SMBus Protocol Summary

The following is a summary of the SMBus protocol. It applies to
all revisions of the protocol (1.0, 1.1, and 2.0).
Certain protocol features which are not supported by
this package are briefly described at the end of this document.

Some adapters understand only the SMBus (System Management Bus) protocol,
which is a subset from the I2C protocol. Fortunately, many devices use
only the same subset, which makes it possible to put them on an SMBus.

If you write a driver for some I2C device, please try to use the SMBus
commands if at all possible (if the device uses only that subset of the
I2C protocol). This makes it possible to use the device driver on both
SMBus adapters and I2C adapters (the SMBus command set is automatically
translated to I2C on I2C adapters, but plain I2C commands can not be
handled at all on most pure SMBus adapters).

Below is a list of SMBus protocol operations, and the functions executing
them. Note that the names used in the SMBus protocol specifications usually
don't match these function names. For some of the operations which pass a
single data byte, the functions using SMBus protocol operation names execute
a different protocol operation entirely.

Key to symbols

```
S     (1 bit) : Start bit
P     (1 bit) : Stop bit
Rd/Wr (1 bit) : Read/Write bit. Rd equals 1, Wr equals 0.
A, NA (1 bit) : Accept and reverse accept bit.
Addr  (7 bits): I2C 7 bit address. Note that this can be expanded as usual to
                get a 10 bit I2C address.
Comm  (8 bits): Command byte, a data byte which often selects a register on
                the device.
Data  (8 bits): A plain data byte. Sometimes, I write DataLow, DataHigh
                for 16 bit data.
Count (8 bits): A data byte containing the length of a block operation.

[..]: Data sent by I2C device, as opposed to data sent by the host adapter.
```

SMBus Quick Command

This sends a single bit to the device, at the place of the Rd/Wr bit.

```
A Addr Rd/Wr [A] P
```

SMBus Receive Byte: i2c_smbus_read_byte()

This reads a single byte from a device, without specifying a device

register. Some devices are so simple that this interface is enough; for
others, it is a shorthand if you want to read the same register as in
the previous SMBus command.

S Addr Rd [A] [Data] NA P

SMBus Send Byte: i2c_smbus_write_byte()
==

This operation is the reverse of Receive Byte: it sends a single byte
to a device. See Receive Byte for more information.

S Addr Wr [A] Data [A] P

SMBus Read Byte: i2c_smbus_read_byte_data()
==

This reads a single byte from a device, from a designated register.
The register is specified through the Comm byte.

S Addr Wr [A] Comm [A] S Addr Rd [A] [Data] NA P

SMBus Read Word: i2c_smbus_read_word_data()
==

This operation is very like Read Byte; again, data is read from a
device, from a designated register that is specified through the Comm
byte. But this time, the data is a complete word (16 bits).

S Addr Wr [A] Comm [A] S Addr Rd [A] [DataLow] A [DataHigh] NA P

SMBus Write Byte: i2c_smbus_write_byte_data()
==

This writes a single byte to a device, to a designated register. The
register is specified through the Comm byte. This is the opposite of
the Read Byte operation.

S Addr Wr [A] Comm [A] Data [A] P

SMBus Write Word: i2c_smbus_write_word_data()
==

This is the opposite of the Read Word operation. 16 bits
of data is written to a device, to the designated register that is
specified through the Comm byte.

S Addr Wr [A] Comm [A] DataLow [A] DataHigh [A] P

SMBus Process Call: i2c_smbus_process_call()

This command selects a device register (through the Comm byte), sends
16 bits of data to it, and reads 16 bits of data in return.

```
S Addr Wr [A] Comm [A] DataLow [A] DataHigh [A]
                            S Addr Rd [A] [DataLow] A [DataHigh] NA P
```

SMBus Block Read: i2c_smbus_read_block_data()

This command reads a block of up to 32 bytes from a device, from a
designated register that is specified through the Comm byte. The amount
of data is specified by the device in the Count byte.

```
S Addr Wr [A] Comm [A]
           S Addr Rd [A] [Count] A [Data] A [Data] A ... A [Data] NA P
```

SMBus Block Write: i2c_smbus_write_block_data()

The opposite of the Block Read command, this writes up to 32 bytes to
a device, to a designated register that is specified through the
Comm byte. The amount of data is specified in the Count byte.

```
S Addr Wr [A] Comm [A] Count [A] Data [A] Data [A] ... [A] Data [A] P
```

SMBus Block Write - Block Read Process Call

SMBus Block Write - Block Read Process Call was introduced in
Revision 2.0 of the specification.

This command selects a device register (through the Comm byte), sends
1 to 31 bytes of data to it, and reads 1 to 31 bytes of data in return.

```
S Addr Wr [A] Comm [A] Count [A] Data [A] ...
                            S Addr Rd [A] [Count] A [Data] ... A P
```

SMBus Host Notify

This command is sent from a SMBus device acting as a master to the
SMBus host acting as a slave.
It is the same form as Write Word, with the command code replaced by the
alerting device's address.

```
[S] [HostAddr] [Wr] A [DevAddr] A [DataLow] A [DataHigh] A [P]
```

Packet Error Checking (PEC)

Packet Error Checking was introduced in Revision 1.1 of the specification.

PEC adds a CRC-8 error-checking byte to transfers using it, immediately
before the terminating STOP.

Address Resolution Protocol (ARP)

The Address Resolution Protocol was introduced in Revision 2.0 of
the specification. It is a higher-layer protocol which uses the
messages above.

ARP adds device enumeration and dynamic address assignment to
the protocol. All ARP communications use slave address 0x61 and
require PEC checksums.

SMBus Alert

SMBus Alert was introduced in Revision 1.0 of the specification.

The SMBus alert protocol allows several SMBus slave devices to share a
single interrupt pin on the SMBus master, while still allowing the master
to know which slave triggered the interrupt.

This is implemented the following way in the Linux kernel:
* I2C bus drivers which support SMBus alert should call
  i2c_setup_smbus_alert() to setup SMBus alert support.
* I2C drivers for devices which can trigger SMBus alerts should implement
  the optional alert() callback.

I2C Block Transactions

The following I2C block transactions are supported by the
SMBus layer and are described here for completeness.
They are *NOT* defined by the SMBus specification.

I2C block transactions do not limit the number of bytes transferred
but the SMBus layer places a limit of 32 bytes.

I2C Block Read: i2c_smbus_read_i2c_block_data()

This command reads a block of bytes from a device, from a
designated register that is specified through the Comm byte.

```
S Addr Wr [A] Comm [A]
          S Addr Rd [A] [Data] A [Data] A ... A [Data] NA P
```

I2C Block Read (2 Comm bytes)

This command reads a block of bytes from a device, from a
designated register that is specified through the two Comm bytes.

```
S Addr Wr [A] Comm1 [A] Comm2 [A]
          S Addr Rd [A] [Data] A [Data] A ... A [Data] NA P
```

I2C Block Write: i2c_smbus_write_i2c_block_data()

The opposite of the Block Read command, this writes bytes to
a device, to a designated register that is specified through the
Comm byte. Note that command lengths of 0, 2, or more bytes are
supported as they are indistinguishable from data.

```
S Addr Wr [A] Comm [A] Data [A] Data [A] ... [A] Data [A] P
```