NAME

smbmsg - send or receive messages over an SMBus

SYNOPSIS

smbmsg [**-f** *dev*] **-p**

smbmsg [**-f** *dev*] **-s** *slave* [**-F** *fmt*] [**-c** *cmd*] [**-w**] [**-i** *incnt*] [**-o** *outcnt*] [*outdata* ...]

DESCRIPTION

The **smbmsg** utility can be used to send or receive messages over an SMBus, see smbus(4).

The **smbmsg** utility has two different modi of operation. The first form shown in the synopsis can be used to "probe" the devices on the SMBus. This is done by sending each valid device address one receive byte, and one quick read message, respectively. Devices that respond to these requests will be displayed by their device address, followed by the strings ‘r’, ‘w’, or ‘rw’, for devices that are readable, writeable, or both, readable and writeable, respectively. The only valid additional option for this modus of operation (besides the **-p** option that chooses the modus) is **-f** *dev*. See below for a description.

Note that probing the bus is risky, since individual devices could perform unwanted actions upon receiving one of the mentioned messages. For example, if a particular SMBus device considers *any* write operation issued to it as a request to power off the system, the probing would trigger this action.

The second form shown in the synopsis can be used to send or receive arbitrary messages to or from individual devices. This might be useful to explore individual devices on the SMBus, or maybe even to write short shell scripts performing maintenance operations on the bus.

Any data values on the command-line are integer values in the range 0 through 255 for byte values, or 0 through 65535 for word values. They can be specified using standard ‘C’ notation (prefix 0 for octal interpretation, or 0x for hexadecimal interpretation).

Since the low-order bit of the device address of SMBus devices selects between read and write operations, only even-numbered slave addresses can exist on the bus.

The options are as follows:

-F *fmt* Specify the printf(3) format to be used for displaying input data. This option is ignored in messages that do not read any input from the SMBus device. The format defaults to ‘0x%02x’ for byte input operations, and to ‘0x%04x’ for word input operations. For multi-byte input (block read), the same format is used for each individual byte read from the SMBus.

-c *cmd* This is the value of the *command* byte to be issued as part of the SMBus message.

-f *dev* This specifies that *dev* should be used as the connection to the SMBus, rather than the default of */dev/smb0*.

-i *incnt* An SMBus message should be generated to read *incnt* bytes from the device.

-o *outcnt* An SMBus message should be generated to write *outcnt* bytes to the device. The data values to write are expected to follow all of the options (and their arguments) on the command-line, where the number of data bytes must match the *outcnt* value.

-p This selects the *probe bus* modus of operation.

-s *slave* The *slave* parameter specifies which SMBus device to connect to. This option also selects the *transfer messages from/to device* modus of operation, where a slave address is mandatory.

-w This option specifies that IO operations are word operations, rather than byte operations. Either *incnt*, or *outcnt* (or both) must be equal 2 in this case. Note that the SMBus byte order is defined to be little-endian (low byte first, high byte follows).

Not all argument combinations make sense in order to form valid SMBus messages. If no **-c** *cmd* option has been provided, the following messages can be issued:

message	**incnt**	**outcnt**
quick read	0	-
quick write	-	0
receive byte	1	-
send byte	-	1

Note in particular that specifying 0 as a count value has a different meaning than omitting the respective option entirely.

If a command value has been given using the **-c** *cmd* option, the following messages can be generated:

message	**-w**	**incnt**	**outcnt**
read byte	no	1	-

write byte	no	-	1
read word	yes	2	-
write word	yes	-	2
process call	yes	2	2
block read	no	>= 2	-
block write	no	-	>= 2

FILES

/dev/smb0 The default device to connect to, unless **-f** *dev* has been provided.

EXIT STATUS

Exit status is 0 on success, or according to sysexits(3) in case of failure.

EXAMPLES

Typical usage examples of the **smbmsg** command include:

```
smbmsg -f /dev/smb1 -p
```

Probe all devices on the SMBus attached to */dev/smb1*.

```
smbmsg -s 0x70 -i 1
```

Issue a *receive byte* message to the device at address 0x70, and display the received byte using the default format.

```
smbmsg -s 0x70 -c 0xff -i 1 -F %d
```

Issue a *read byte* message to the device at slave address 0x70, using 255 (0xff) as the command-byte to send to the device, and display the result using the custom format ‘%d’.

```
smbmsg -s 0xa0 -c 0 -o 1 0x80
```

Send a *write byte* message to the slave device at address 0xa0, using 0 as the command-byte value, and 0x80 as the byte to send (after the command). Assuming this might be a Philips PCF8583 real-time clock, this would stop the clock.

```
smbmsg -s 0xa0 -c 1 -i 6 -F %02x
```

Send a *block read* command to device at address 0xa0, and read 6 bytes from it, using hexadecimal display. Again, assuming a PCF8583 RTC, this would display the fractions of second, seconds, minutes, hours, year/date, and weekday/month values. Since this RTC uses BCD notation, the actual values displayed were decimal then.

```
smbmsg -s 0xa0 -c 2 -o 5 0x00 0x07 0x22 0x16 0x05
```

Send a *block write* command to device at address 0xa0. For the PCF8583 RTC, this would set the clock to Sunday (2004%4)-05-16 22:07:00.

DIAGNOSTICS

Diagnostic messages issued are supposed to be self-explanatory.

SEE ALSO

printf(3), sysexits(3), smb(4), smbus(4)

The SMBus specification, http://www.smbus.org/specs/.

HISTORY

The **smbmsg** utility first appeared in FreeBSD 5.3.

AUTHORS

The **smbmsg** utility and this manual page were written by Jörg Wunsch.