NAME

i2c - test I2C bus and slave devices

SYNOPSIS

i2c -a *address* [**-f** *device*] [**-d** *r|w*] [**-w** *0|8|16|16LE|16BE*] [**-o** *offset*] [**-c** *count*] [**-m** *tr|ss|rs|no*] [**-b**] [**-v**]
i2c -h
i2c -i [**-v**] [*cmd ...*] [-]
i2c -r [**-f** *device*] [**-v**]
i2c -s [**-f** *device*] [**-n** *skip_addr*] [**-v**]

DESCRIPTION

The **i2c** utility can be used to perform raw data transfers (read or write) to devices on an I2C bus. It can also scan the bus for available devices and reset the I2C controller.

The options are as follows:

-a *address* 7-bit address on the I2C device to operate on (hex).

-b binary mode - when performing a read operation, the data read from the device is output in binary format on stdout.

-c *count* number of bytes to transfer (decimal).

-d *r|w* transfer direction: r - read, w - write. Data to be written is read from stdin as binary bytes.

-f *device* I2C bus to use (default is /dev/iic0).

-i Interpreted mode

-h Help

-m *tr|ss|rs|no* addressing mode, i.e., I2C bus operations performed after the offset for the transfer has been written to the device and before the actual read/write operation.

- *tr* complete-transfer
- *ss* stop then start
- *rs* repeated start
- *no* none

Some I2C bus hardware does not provide control over the individual start, repeat-start, and stop operations. Such hardware can only perform a complete transfer of the offset

and the data as a single operation. The *tr* mode creates control structures describing the transfer and submits them to the driver as a single complete transaction. This mode works on all types of I2C hardware.

-n *skip_addr* address(es) to be skipped during bus scan. One or more addresses ([0x]xx) or ranges of addresses ([0x]xx-[0x]xx or [0x]xx..[0x]xx) separated by commas or colons.

-o *offset* offset within the device for data transfer (hex). The default is zero. Use "-w 0" to disable writing of the offset to the slave.

-r reset the controller.

-s scan the bus for devices.

-v be verbose.

-w *0|8|16|16LE|16BE*
device offset width (in bits). This is used to determine how to pass *offset* specified with **-o** to the slave. Zero means that the offset is ignored and not passed to the slave at all. The endianness defaults to little-endian.

INTERPRETED MODE

When started with **-i** any remaining arguments are interpreted as commands, and if the last argument is '-', or there are no arguments, commands will (also) be read from stdin.

Available commands:

'r' bus address [0|8|16|16LE|16BE] offset count
Read command, count bytes are read and hexdumped to stdout.

'w' bus address [0|8|16|16LE|16BE] offset hexstring
Write command, hexstring (white-space is allowed) is written to device.

'p' anything
Print command, the entire line is printed to stdout. (This can be used for synchronization.)

All numeric fields accept canonical decimal/octal/hex notation.

Without the **-v** option, all errors are fatal with non-zero exit status.

With the **-v** option, no errors are fatal, and all commands will return either "OK\n" or "ERROR\n" on stdout. In case of error, detailed diagnostics will precede that on stderr.

Blank lines and lines starting with '#' are ignored.

EXAMPLES

- Scan the default bus (/dev/iic0) for devices:

  i2c -s

- Scan the default bus (/dev/iic0) for devices and skip addresses 0x45 to 0x47 (inclusive) and 0x56.

  i2c -s -n 0x56,45-47

- Read 8 bytes of data from device at address 0x56 (e.g., an EEPROM):

  i2c -a 0x56 -d r -c 8

- Write 16 bytes of data from file data.bin to device 0x56 at offset 0x10:

  i2c -a 0x56 -d w -c 16 -o 0x10 -b < data.bin

- Copy 4 bytes between two EEPROMs (0x56 on /dev/iic1 to 0x57 on /dev/iic0):

  i2c -a 0x56 -f /dev/iic1 -d r -c 0x4 -b | i2c -a 0x57 -f /dev/iic0 -d w -c 4 -b

- Reset the controller:

  i2c -f /dev/iic1 -r

- Read 8 bytes at address 24 in an EEPROM:

  i2c -i 'r 0 0x50 16BE 24 8'

- Read 2x8 bytes at address 24 and 48 in an EEPROM:

  echo 'r 0 0x50 16BE 48 8' | i2c -i 'r 0 0x50 16BE 24 8' -

WARNING

Many systems store critical low-level information in I2C memories, and may contain other I2C devices, such as temperature or voltage sensors. Reading these can disturb the firmware's operation and writing to them can "brick" the hardware.

SEE ALSO

iic(4), iicbus(4) smbus(4)

HISTORY

The **i2c** utility appeared in FreeBSD 8.0.

AUTHORS

The **i2c** utility and this manual page were written by Bartlomiej Sieka <*tur@semihalf.com*> and Michal Hajduk <*mih@semihalf.com*>.

Poul-Henning Kamp <*phk@FreeBSD.org*> added interpreted mode.