# Bash Shell Interface for I2C

On the kernel level, I2C communication is handled by the `i2c-dev` kernel driver. On the lowest level, the bus is seen as a file and using a file descriptor one can read from it or write to it. The shell interface is a set of compiled programs that wrap this functionality for specific uses at a higher level. The intended use is interactive debugging to see if something works or not. For programmatic use it's better to take a look at the C API for C/C++ and the smbus2 library for Python.

These programs are part of the `i2c-tools` package that usually comes preinstalled in most Linux distributions.

## i2cdetect

Detects I2C chips. Use: `i2cdetect [options] i2cbus [first_addr last_addr]`. Special flags that don't require a given i2cbus are `-V` that prints the current version and `-l`.

- `i2cdetect -l` to list busses. To access each buss you have to used it's number (`i2c-{NUM}`) or the full string name.
- `-q` and `-r` options are to use quick write and receive byte commands for probing. They can corrupt certain IC chips.
- `-F` displays the list of functionalities available on the bus.

## i2cdump

Examines registers on the I2C bus. Use: `i2cdump [options] i2cbus address [mode [bank [bankreg]]]`

- `i2cbus`: One of the buses represented by the number or string name from `i2cdetect -l`.
- `-r first-last`: Limit the range of registers being accessed. Incompatible with `s`.
- `address`: The address of the IC chip on the bus.
- `mode`: Chooses the mode to access the register as a single byte, a double byte word or a block. If not provided, defaults to single byte without PEC. Options are listed below:
  - `b`: Single byte (default).
  - `w`: Word.
  - `i`: I2C block.
  - `c`: Consecutive block. Useful for ICs with automatic increment, like EEPROMs.
  - `W`: Word, but the command will only be issued on even addresses. Also useful for EEPROMs.
  - `s`: SMBus block. Avoid it, it's deprecated.
  - `p`: Can be appended on the mode parameter (not on `i` and `W`) to enable Packet Error Checking (PEC).
- `bank`/`bankreg`: Used on Winbond and Asus chips. `bank` is an integer from 0 to 7, and `bankreg` an integer from 0x00 to 0xff with a default of 0x4e.

Use of `i2cdump` is not advised unless you have intimate knowledge of the IC you're accessing. It's very easy to corrupt stuff.

# i2cget

3

# i2cset

4

# i2ctransfer

Send multiple I2C messages/bytes in one transfer. Use `i2ctransfer [options] i2cbus desc [data] [desc [data]] …`.

Options:

- `-v`: Enable verbose output for all messages (both read and write).
- `-y`: Don't wait for user to interactively confirm the communication.
- `-b`: Print read messages as raw binary data. Can be used to save to file or to pipe into tools like hexdump for beautiful printing. Disables the verbose option.

Arguments:

- `i2cbus`: The bus number or name, as listed in `i2cdetect -l`.
- `desc` blocks: Block describing the message. `{r|w}n[@address]`. `r` or `w` describe if it's a read or a write message. `n` is the number of bytes read or written in this message, with an upper limit of 8192 due to the linux kernel. If reading from an IC with SMBus Block support, `?` can be used to let the slave determine the number of bytes. The `@address` part is required for the first message, and is simply the address of the IC on the I2C bus. For messages without the address part, it is implicitly assumed to be the same as the previous message. Addresses can only be 7bit, there's no support for 10bit addressing.
- `data`: If the message is a write, the actual bytes as numeric values make up the data argument. Hex, octal and decimal are supported generally, with binary support depending on the tool version. The values can be explicitly written one by one (`0xDE 0xAD 0xBE 0xEF`), or a suffix can be used. The suffixes are:
  - `=`: `3=` means `3 3 3 3 3…`
  - `+`: `0x0d+` means `0x0d 0x0e 0x0f 0x10 0x11…`
  - `-`: `0xff-` means `0xff 0xfe 0xfd 0xfc 0xfb…`
- Additional `[desc [data]]` pairs can be chained. Each one is called an I2C Block. The upper limit defined by the linux kernel is 42 at the time of writing.

Examples:

- `i2ctransfer 5 w1@0x73 0x20 r24 w1@0x74 0x20 r16`: This is an example of reading from 2 EEPROM ICs. First we set the internal pointer of EEPROM 1 at address 0x73 to read from point 0x20, and then we read 24 bytes. Then we read 16 bytes starting from 0x20 on EEPROM 2

on address 0x74. Both EEPROMS are on bus 5, meaning /dev/i2c-5.

- `i2ctransfer 8 w1@0x40 0x17 r2 w1 0x27 r2`: This is an example of reading values from an ADC on but i2c-8 at address 0x40. First we ask to read from a channel through the control code 0x17 = 0b0001_0111. This could mean read from channel 1 with settings false, true, true, true. Then we read a 14-bit input, which requires 2 bytes for transfer. Then we sent the control code 0x27=0b0010_0111 which could mean read from channel 2. Then again we read the 14-bit input with 2 bytes of transfer.

From:
https://wiki.plakatouras.com/ - **Personal Wiki**

Permanent link:
**https://wiki.plakatouras.com/doku.php?id=documentation:protocols:i2c:i2c-tools_shell**

Last update: **2026/10/12 13:08**