MODULE: i2c-stub

DESCRIPTION:

This module is a very simple fake I2C/SMBus driver. It implements five types of SMBus commands: write quick, (r/w) byte, (r/w) byte data, (r/w) word data, and (r/w) I2C block data.

You need to provide chip addresses as a module parameter when loading this driver, which will then only react to SMBus commands to these addresses.

No hardware is needed nor associated with this module. It will accept write quick commands to the specified addresses; it will respond to the other commands (also to the specified addresses) by reading from or writing to arrays in memory. It will also spam the kernel logs for every command it handles.

A pointer register with auto-increment is implemented for all byte operations. This allows for continuous byte reads like those supported by EEPROMs, among others.

The typical use-case is like this:

1. load this module
2. use i2cset (from the i2c-tools project) to pre-load some data
3. load the target chip driver module
4. observe its behavior in the kernel log

There's a script named i2c-stub-from-dump in the i2c-tools package which can load register values automatically from a chip dump.

PARAMETERS:

int chip_addr[10]:
    The SMBus addresses to emulate chips at.

unsigned long functionality:
    Functionality override, to disable some commands. See I2C_FUNC_* constants in <linux/i2c.h> for the suitable values. For example, value 0x1f0000 would only enable the quick, byte and byte data commands.

CAVEATS:

If your target driver polls some byte or word waiting for it to change, the stub could lock it up. Use i2cset to unlock it.

If the hardware for your driver has banked registers (e.g. Winbond sensors chips) this module will not work well - although it could be extended to support that pretty easily.

If you spam it hard enough, printk can be lossy. This module really wants something like relayfs.