I2C device driver binding control from user-space in old kernels

> **Note**
>
> Note: this section is only relevant if you are handling some old code found in kernel 2.6. If you work with more recent kernels, you can safely skip this section.

Up to kernel 2.6.32, many I2C drivers used helper macros provided by <linux/i2c.h> which created standard module parameters to let the user control how the driver would probe I2C buses and attach to devices. These parameters were known as `probe` (to let the driver probe for an extra address), `force` (to forcibly attach the driver to a given device) and `ignore` (to prevent a driver from probing a given address).

With the conversion of the I2C subsystem to the standard device driver binding model, it became clear that these per-module parameters were no longer needed, and that a centralized implementation was possible. The new, sysfs-based interface is described in Documentation/i2c/instantiating-devices.rst, section "Method 4: Instantiate from user-space".

Below is a mapping from the old module parameters to the new interface.

Attaching a driver to an I2C device

Old method (module parameters):

```
# modprobe <driver> probe=1,0x2d
# modprobe <driver> force=1,0x2d
# modprobe <driver> force_<device>=1,0x2d
```

New method (sysfs interface):

```
# echo <device> 0x2d > /sys/bus/i2c/devices/i2c-1/new_device
```

Preventing a driver from attaching to an I2C device

Old method (module parameters):

```
# modprobe <driver> ignore=1,0x2f
```

New method (sysfs interface):

```
# echo dummy 0x2f > /sys/bus/i2c/devices/i2c-1/new_device
# modprobe <driver>
```

Of course, it is important to instantiate the `dummy` device before loading the driver. The dummy device will be handled by i2c-core itself, preventing other drivers from binding to it later on. If there is a real device at the problematic address, and you want another driver to bind to it, then simply pass the name of the device in question instead of `dummy`.