# Kernel driver i2c-i801

Supported adapters:

- Intel 82801AA and 82801AB (ICH and ICH0 - part of the '810' and '810E' chipsets)
- Intel 82801BA (ICH2 - part of the '815E' chipset)
- Intel 82801CA/CAM (ICH3)
- Intel 82801DB (ICH4) (HW PEC supported)
- Intel 82801EB/ER (ICH5) (HW PEC supported)
- Intel 6300ESB
- Intel 82801FB/FR/FW/FRW (ICH6)
- Intel 82801G (ICH7)
- Intel 631xESB/632xESB (ESB2)
- Intel 82801H (ICH8)
- Intel 82801I (ICH9)
- Intel EP80579 (Tolapai)
- Intel 82801JI (ICH10)
- Intel 5/3400 Series (PCH)
- Intel 6 Series (PCH)
- Intel Patsburg (PCH)
- Intel DH89xxCC (PCH)
- Intel Panther Point (PCH)
- Intel Lynx Point (PCH)
- Intel Avoton (SOC)
- Intel Wellsburg (PCH)
- Intel Coleto Creek (PCH)
- Intel Wildcat Point (PCH)
- Intel BayTrail (SOC)
- Intel Braswell (SOC)
- Intel Sunrise Point (PCH)
- Intel Kaby Lake (PCH)
- Intel DNV (SOC)
- Intel Broxton (SOC)
- Intel Lewisburg (PCH)
- Intel Gemini Lake (SOC)
- Intel Cannon Lake (PCH)
- Intel Cedar Fork (PCH)
- Intel Ice Lake (PCH)
- Intel Comet Lake (PCH)
- Intel Elkhart Lake (PCH)
- Intel Tiger Lake (PCH)
- Intel Jasper Lake (SOC)
- Intel Emmitsburg (PCH)
- Intel Alder Lake (PCH)
- Intel Raptor Lake (PCH)

Datasheets: Publicly available at the Intel website

On Intel Patsburg and later chipsets, both the normal host SMBus controller and the additional 'Integrated Device Function' controllers are supported.

Authors:
- Mark Studebaker <mdsxyz123@yahoo.com>
- Jean Delvare <jdelvare@suse.de>

## Module Parameters

- disable_features (bit vector)

Disable selected features normally supported by the device. This makes it possible to work around possible driver or hardware bugs if the feature in question doesn't work as intended for whatever reason. Bit values:

| | |
|---|---|
| 0x01 | disable SMBus PEC |
| 0x02 | disable the block buffer |
| 0x08 | disable the I2C block read functionality |

| | |
|---|---|
| 0x10 | don't use interrupts |
| 0x20 | disable SMBus Host Notify |

## Description

The ICH (properly known as the 82801AA), ICH0 (82801AB), ICH2 (82801BA), ICH3 (82801CA/CAM) and later devices (PCH) are Intel chips that are a part of Intel's '810' chipset for Celeron-based PCs, '810E' chipset for Pentium-based PCs, '815E' chipset, and others.

The ICH chips contain at least SEVEN separate PCI functions in TWO logical PCI devices. An output of lspci will show something similar to the following:

```
00:1e.0 PCI bridge: Intel Corporation: Unknown device 2418 (rev 01)
00:1f.0 ISA bridge: Intel Corporation: Unknown device 2410 (rev 01)
00:1f.1 IDE interface: Intel Corporation: Unknown device 2411 (rev 01)
00:1f.2 USB Controller: Intel Corporation: Unknown device 2412 (rev 01)
00:1f.3 Unknown class [0c05]: Intel Corporation: Unknown device 2413 (rev 01)
```

The SMBus controller is function 3 in device 1f. Class 0c05 is SMBus Serial Controller.

The ICH chips are quite similar to Intel's PIIX4 chip, at least in the SMBus controller.

## Process Call Support

Block process call is supported on the 82801EB (ICH5) and later chips.

## I2C Block Read Support

I2C block read is supported on the 82801EB (ICH5) and later chips.

## SMBus 2.0 Support

The 82801DB (ICH4) and later chips support several SMBus 2.0 features.

## Interrupt Support

PCI interrupt support is supported on the 82801EB (ICH5) and later chips.

## Hidden ICH SMBus

If your system has an Intel ICH south bridge, but you do NOT see the SMBus device at 00:1f.3 in lspci, and you can't figure out any way in the BIOS to enable it, it means it has been hidden by the BIOS code. Asus is well known for first doing this on their P4B motherboard, and many other boards after that. Some vendor machines are affected as well.

The first thing to try is the "i2c-scmi" ACPI driver. It could be that the SMBus was hidden on purpose because it'll be driven by ACPI. If the i2c-scmi driver works for you, just forget about the i2c-i801 driver and don't try to unhide the ICH SMBus. Even if i2c-scmi doesn't work, you better make sure that the SMBus isn't used by the ACPI code. Try loading the "fan" and "thermal" drivers, and check in /sys/class/thermal. If you find a thermal zone with type "acpitz", it's likely that the ACPI is accessing the SMBus and it's safer not to unhide it. Only once you are certain that ACPI isn't using the SMBus, you can attempt to unhide it.

In order to unhide the SMBus, we need to change the value of a PCI register before the kernel enumerates the PCI devices. This is done in drivers/pci/quirks.c, where all affected boards must be listed (see function asus_hides_smbus_hostbridge.) If the SMBus device is missing, and you think there's something interesting on the SMBus (e.g. a hardware monitoring chip), you need to add your board to the list.

The motherboard is identified using the subvendor and subdevice IDs of the host bridge PCI device. Get yours with `lspci -n -v -s 00:00.0`:

```
00:00.0 Class 0600: 8086:2570 (rev 02)
        Subsystem: 1043:80f2
        Flags: bus master, fast devsel, latency 0
        Memory at fc000000 (32-bit, prefetchable) [size=32M]
        Capabilities: [e4] #09 [2106]
        Capabilities: [a0] AGP version 3.0
```

Here the host bridge ID is 2570 (82865G/PE/P), the subvendor ID is 1043 (Asus) and the subdevice ID is 80f2 (P4P800-X). You can find the symbolic names for the bridge ID and the subvendor ID in include/linux/pci_ids.h, and then add a case for your subdevice ID at the right place in drivers/pci/quirks.c. Then please give it very good testing, to make sure that the unhidden SMBus doesn't conflict with e.g. ACPI.

If it works, proves useful (i.e. there are usable chips on the SMBus) and seems safe, please submit a patch for inclusion into the

kernel.

Note: There's a useful script in lm_sensors 2.10.2 and later, named unhide_ICH_SMBus (in prog/hotplug), which uses the fakephp driver to temporarily unhide the SMBus without having to patch and recompile your kernel. It's very convenient if you just want to check if there's anything interesting on your hidden ICH SMBus.

---

The lm_sensors project gratefully acknowledges the support of Texas Instruments in the initial development of this driver.

The lm_sensors project gratefully acknowledges the support of Intel in the development of SMBus 2.0 / ICH4 features of this driver.