

Driver changes

This file details changes in 2.6 which affect PCMCIA card driver authors:

- `pcmcia_loop_config()` and `autoconfiguration` (as of 2.6.36)

If `struct pcmcia_device *p_dev->config_flags` is set accordingly, `pcmcia_loop_config()` now sets up certain configuration values automatically, though the driver may still override the settings in the callback function. The following `autoconfiguration` options are provided at the moment:

- `CONF_AUTO_CHECK_VCC` : check for matching Vcc
- `CONF_AUTO_SET_VPP` : set Vpp
- `CONF_AUTO_AUDIO` : auto-enable audio line, if required
- `CONF_AUTO_SET_IO` : set ioport resources (`->resource[0,1]`)
- `CONF_AUTO_SET_IOMEM` : set first iomem resource (`->resource[2]`)

- `pcmcia_request_configuration -> pcmcia_enable_device` (as of 2.6.36)

`pcmcia_request_configuration()` got renamed to `pcmcia_enable_device()`, as it mirrors `pcmcia_disable_device()`. Configuration settings are now stored in `struct pcmcia_device`, e.g. in the fields `config_flags`, `config_index`, `config_base`, `vpp`.

- `pcmcia_request_window` changes (as of 2.6.36)

Instead of `win_req_t`, drivers are now requested to fill out `struct pcmcia_device *p_dev->resource[2,3,4,5]` for up to four ioport ranges. After a call to `pcmcia_request_window()`, the regions found there are reserved and may be used immediately -- until `pcmcia_release_window()` is called.

- `pcmcia_request_io` changes (as of 2.6.36)

Instead of `io_req_t`, drivers are now requested to fill out `struct pcmcia_device *p_dev->resource[0,1]` for up to two ioport ranges. After a call to `pcmcia_request_io()`, the ports found there are reserved, after calling `pcmcia_request_configuration()`, they may be used.

- No `dev_info_t`, no `cs_types.h` (as of 2.6.36)

`dev_info_t` and a few other typedefs are removed. No longer use them in PCMCIA device drivers. Also, do not include `pcmcia/cs_types.h`, as this file is gone.

- No `dev_node_t` (as of 2.6.35)

There is no more need to fill out a "dev_node_t" structure.

- New IRQ request rules (as of 2.6.35)

Instead of the old `pcmcia_request_irq()` interface, drivers may now choose between:

- calling `request_irq/free_irq` directly. Use the IRQ from `*p_dev->irq`.
- use `pcmcia_request_irq(p_dev, handler_t)`; the PCMCIA core will clean up automatically on calls to `pcmcia_disable_device()` or device ejection.

- no `cs_error` / `CS_CHECK` / `CONFIG_PCMCIA_DEBUG` (as of 2.6.33)

Instead of the `cs_error()` callback or the `CS_CHECK()` macro, please use Linux-style checking of return values, and -- if necessary -- debug messages using "dev_dbg()" or "pr_debug()".

- New CIS tuple access (as of 2.6.33)

Instead of `pcmcia_get_{first,next}_tuple()`, `pcmcia_get_tuple_data()` and `pcmcia_parse_tuple()`, a driver shall use "pcmcia_get_tuple()" if it is only interested in one (raw) tuple, or "pcmcia_loop_tuple()" if it is interested in all tuples of one type. To decode the MAC from CISTPL_FUNCCE, a new helper "pcmcia_get_mac_from_cis()" was added.

- New configuration loop helper (as of 2.6.28)

By calling `pcmcia_loop_config()`, a driver can iterate over all available configuration options. During a driver's `probe()` phase, one doesn't need to use `pcmcia_get_{first,next}_tuple`, `pcmcia_get_tuple_data` and `pcmcia_parse_tuple` directly in most if not all cases.

- New release helper (as of 2.6.17)

Instead of calling `pcmcia_release_{configuration,io,irq,win}`, all that's necessary now is calling `pcmcia_disable_device`. As there is no valid reason left to call `pcmcia_release_io` and `pcmcia_release_irq`, the exports for them were removed.

- Unify detach and REMOVAL event code, as well as attach and INSERTION code (as of 2.6.16):

```
void (*remove)          (struct pcmcia_device *dev);
int (*probe)            (struct pcmcia_device *dev);
```

- Move suspend, resume and reset out of event handler (as of 2.6.16):

```
int (*suspend)          (struct pcmcia_device *dev);
int (*resume)           (struct pcmcia_device *dev);
```

should be initialized in struct `pcmcia_driver`, and handle (`SUSPEND == RESET_PHYSICAL`) and (`RESUME == CARD_RESET`) events

- event handler initialization in struct `pcmcia_driver` (as of 2.6.13)
The event handler is notified of all events, and must be initialized as the `event()` callback in the driver's struct `pcmcia_driver`.
- `pcmcia/version.h` should not be used (as of 2.6.13)
This file will be removed eventually.
- in-kernel device<->driver matching (as of 2.6.13)
PCMCIA devices and their correct drivers can now be matched in kernelspace. See 'devicetable.txt' for details.
- Device model integration (as of 2.6.11)
A struct `pcmcia_device` is registered with the device model core, and can be used (e.g. for `SET_NETDEV_DEV`) by using `handle_to_dev(client_handle_t * handle)`.
- Convert internal I/O port addresses to unsigned int (as of 2.6.11)
`ioaddr_t` should be replaced by unsigned int in PCMCIA card drivers.
- `irq_mask` and `irq_list` parameters (as of 2.6.11)
The `irq_mask` and `irq_list` parameters should no longer be used in PCMCIA card drivers. Instead, it is the job of the PCMCIA core to determine which IRQ should be used. Therefore, `link->irq.IRQInfo2` is ignored.
- `client->PendingEvents` is gone (as of 2.6.11)
`client->PendingEvents` is no longer available.
- `client->Attributes` are gone (as of 2.6.11)
`client->Attributes` is unused, therefore it is removed from all PCMCIA card drivers
- core functions no longer available (as of 2.6.11)
The following functions have been removed from the kernel source because they are unused by all in-kernel drivers, and no external driver was reported to rely on them:

```
pcmcia_get_first_region()
pcmcia_get_next_region()
pcmcia_modify_window()
pcmcia_set_event_mask()
pcmcia_get_first_window()
pcmcia_get_next_window()
```

- device list iteration upon module removal (as of 2.6.10)
It is no longer necessary to iterate on the driver's internal client list and call the `->detach()` function upon module removal.
- Resource management. (as of 2.6.8)
Although the PCMCIA subsystem will allocate resources for cards, it no longer marks these resources busy. This means that driver authors are now responsible for claiming your resources as per other drivers in Linux. You should use `request_region()` to mark your IO regions in-use, and `request_mem_region()` to mark your memory regions in-use. The name argument should be a pointer to your driver name. Eg. for `pcnet_cs`, name should point to the string "pcnet_cs".
- CardServices is gone `CardServices()` in 2.4 is just a big switch statement to call various services. In 2.6, all of those entry points are exported and called directly (except for `pcmcia_report_error()`, just use `cs_error()` instead).
- struct `pcmcia_driver` You need to use struct `pcmcia_driver` and `pcmcia_{un,}register_driver` instead of `{un,}register_pccard_driver`