COSATEQ

CO-PCICAN Linux Device Driver

COSATEQ

CO-PCICAN Linux Device Driver and Low Level API V1.1

Januar 2011

Manual Version1.1

© cosateQ
Seehaldeweg 11 • 88239 Wangen
Telefon +49 (0)7522 9749-0 • Fax +49 (0)7522 9749-49
Mail info@cosateq.de • Web www.cosateq.com

Overview

Software

Handling

Attachment

Chapter 1

Overview

CO-PCICAN

The CO-PCICAN PCI interface card offers 4 CAN-channels. All channels are accessible with a standard Sub-D connector and an optional slot extension. Hardware, firmware and driver are optimized for operation in a real-time system, especially with Cosateq's real-time environment SCALE-RT. The card is also able to handle tasks in other areas of application, e.g. controlling or visualization.

For CANopen there is also a driver interface to the open source CanFestival framework (get latest revision from http://www.canfestival.org). This is not part of this device driver interface.

To handle the CO_PCICAN card in a standard Linux environment a Linux Kernel (>2.6.14) device driver has been created. This document will show you what this driver is able to do and how to use it.

Functional Overview

The driver is divided into two parts.
The low level driver runs in kernel mode and is responsible for the hardware interface. It searches the PCI Interface and initializes all CO-PCICAN cards found. All cards are maintained with the help of a linked list. The second purpose is to channel user space calls via ioctl() to the hardware.
The API runs in user mode. It provides abstraction for all low level tasks, like channel configuration or writing data.

Chapter 2

Software

Driver

For the kernel mode driver, there are only a few interfaces implemented. These are Open, Release and IO Control.

```
struct file_operations co_fops =
{
  owner:        THIS_MODULE,
  ioctl:        co_ioctl,
  open:         co_open,
  release:      co_release
};
```

Open and release are the kernel space callbacks for the open() and free() system calls from user space. IO Control handles all operation with the card as target, e.g. configuring channels, writing data, reading data, ….

API

The API is an abstraction for the kernel mode driver. It should help the user to avoid trouble with system calls.

co_pcican_init

Trys to open a given device file and returns a pointer to a device struct (which is only an abstraction for the file descriptor).

```
/**
 * Gets the device name and returns CO_PCICAN_DEV_t to
 * work with.
 *
 * @param devname path to the device file. e.g.
/dev/co_pcican-0
 * @return NULL for error, != 0 for success
 */
CO_PCICAN_DEV_t* co_pcican_init(const char* devname)
```

co_pcican_free

Closes the file and frees the device struct.

```
/**
```

```
 * closes file handle and frees memory for the device object
 *
 * @param dev device file handler
 * @return < 0 for error, 1 for success
 */
int co_pcican_free(CO_PCICAN_DEV_t* dev)
```

co_pcican_enter_run_mode

Sets the CO-PCICAN card into run mode. In this mode the card only response to read and write commands.

```
/**
 * sets the CO_PCICAN into run mode
 *
 * @param dev device file handler
 * @return -1 for error
 */
int co_pcican_enter_run_mode(CO_PCICAN_DEV_t* dev)
```

co_pcican_enter_config_mode

Sets the CO-PCICAN card into config mode. In ths mode the card response to config commands.

```
/**
 * sets the CO_PCICAN into config mode
 *
 * @param dev device file handler
 * @return -1 for error
 */
int co_pcican_enter_config_mode(CO_PCICAN_DEV_t* dev)
```

co_pcican_enter_loader_mode

Sets the CO-PCICAN card into loader mode. This allows to update the firmware.

```
/**
 * sets the CO_PCICAN into loader mode
 *
 * @param dev device file handler
 * @return -1 for error
 */
int co_pcican_enter_loader_mode(CO_PCICAN_DEV_t* dev)
```

co_pcican_get_mode

Returns the current mode. This mode is not implemented yet.

```
/**
 * returns the mode the board is currently in
 *
 * @param dev device file handler
 * @return -1 for error
 */
```

```
CO_PCICAN_MODE_t co_pcican_get_mode(CO_PCICAN_DEV_t* dev)
```

co_pcican_get_brdtime

Returns the current board time. This time is used to create time stamps for incoming messages. This time is currently not synchronized with the global PC time, so it may differ over time.

```
/**
 * returns the current board time
 *
 * @param dev device file handler
 * @return 64 bit board time
 */
unsigned long long co_pcican_get_brdtime(CO_PCICAN_DEV_t*
dev)
```

co_pcican_config

Configuration for a channel. You can set the speed between 10kBaud and 1000kBaud. Furthermore it is possible to enable or disable the "listen only" mode of the channel.

```
/**
 * sets the config for a given tx channel
 *
 * @param dev device file handler
 * @param channel channel to be configured
 * @param speed 0 - 7 for different speed adjustments
 * @param listenonly 0 for disable; 1 for enable
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int co_pcican_config(CO_PCICAN_DEV_t* dev, unsigned char
channel, unsigned char speed, unsigned char listenonly)
```

co_pcican_write

Writes a CAN message into the sender FIFO of a given channel. To send a CAN Message you must also set the type, size and id parameter.

```
/**
 * writes a message into a given tx channel
 *
 * @param dev device file handler
 * @param channel channel to be configured
 * @param type Type of the CAN message
 * @param size Size of the Message
 * @param id ID of the Message
 * @param message data member of the struct will be sent,
time will be ignored
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int co_pcican_write(CO_PCICAN_DEV_t* dev, unsigned char
channel, unsigned long type, unsigned long size, unsigned
long id, CO_PCICAN_MESSAGE_t* message)
```

co_pcican_read

Reads a CAN Message from the receiver FIFO of a given channel. Returns a -1 if there is nothing to read.

```
/**
 * reads a message from a given tx channel
 *
 * @param dev device file handler
 * @param channel channel to be configured

* @param *type points to the Type of the received MSG
* @param *size points to the Size of the received MSG
* @param *id points to the id of the received msg
* @param message data member of the struct will be the CAN
word, time will be the time stamp
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int   co_pcican_read(CO_PCICAN_DEV_t*   dev,   unsigned   char
channel, unsigned long *type, unsigned long *size, unsigned
long *id, CO_PCICAN_MESSAGE_t* message)
```

co_pcican_timestamp_to_ms

Calculates timestamp value to ms. Returns a timestamp in ms.

```
/**
 * calculates timestamp value to ms
 *
 * @param message data member of the struct will be the can
word, time will be the time stamp
 * @return -EINVAL for invalid parameter or timestamp in ms
 */
double          co_pcican_timestamp_to_ms(CO_PCICAN_MESSAGE_t*
message)
```

Chapter 3

Handling

Dependencies

Kernel sources to compile the driver. !!!Kernel version has to be >2.6.14 !!!

crc16 kernel module is needed.

Loading the driver

You will get all files as a tgz file. Get the latest version from www.cosateq.com. Unpack it with

```
tar –xzvf co_pcican_driver.tgz
```

Then build the driver and the API including the example 'test' with

```
make driver
make api
make test
```

To load the driver, you only have to run the co_canload.sh script from the shell. The script creates a node under /dev. This node should be named co_pcican-X. With the X standing for the number of the board you have plugged in your PC. E.g. with two cards in your PCI, you should have two nodes named co_pciacan-0 and co_pcican-1.

More information about hardware, firmware and I/O addresses can be found in the system log.

To update your firmware if necessary please use the 'co_fw_update' tool.

Example

With the API comes a small example called 'test' for using the API. This little program (written in C) opens a device (co_pcican-0) and sends incremented data packets on channel 0 and tries to catch the same data on rx channel 1. The example assumes that tx channel 0 is connected to rx channel 1 externally.

Chapter 4

Attachment

Related documents

hardware_reference_CO-PCICAN_v0.4 (December 2009)

History

1.0: Initial release

1.1: Modified detach of device and introduce _IOC macro in Linux driver