CO-PCIA429 Linux Device Driver

COSATEQ

CO-PCIA429 Linux Device Driver and Low Level API V1.1

Januar 2011

Manual Version 1.1

© 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

Chapter 1

Overview

CO-PCIA429

The CO-PCIA429 PCI interface card offers up to 16TX + 16RX ARINC429-channels. All channels are independent. 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.
To handle the CO_PCIA429 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-PCIA429 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_pcia429_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_PCIA429_DEV_t to
 * work with.
 *
 * @param devname path to the device file. e.g.
/dev/co_pcia429-0
 * @return NULL for error, != 0 for success
 */
CO_PCIA429_DEV_t* co_pcia429_init(const char* devname)
```

co_pcia429_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_pcia429_free(CO_PCIA429_DEV_t* dev)
```

co_pcia429_enter_run_mode

Sets the CO-PCIA429 card into run mode. In this mode the card only response to read and write commands.

```
/**
 * sets the CO_PCIA429 into run mode
 *
 * @param dev device file handler
 * @return -1 for error
 */
int co_pcia429_enter_run_mode(CO_PCIA429_DEV_t* dev)
```

co_pcia429_enter_config_mode

Sets the CO-PCIA429 card into config mode. In this mode the card response to config commands.

```
/**
 * sets the CO_PCIA429 into config mode
 *
 * @param dev device file handler
 * @return -1 for error
 */
int co_pcia429_enter_config_mode(CO_PCIA429_DEV_t* dev)
```

co_pcia429_enter_loader_mode

Sets the CO-PCIA429 card into loader mode. This allows to update the firmware.

```
/**
 * sets the CO_PCIA429 into loader mode
 *
 * @param dev device file handler
 * @return -1 for error
 */
int co_pcia429_enter_loader_mode(CO_PCIA429_DEV_t* dev)
```

co_pcia429_get_mode

Returns the current mode.

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

```
CO_PCIA429_MODE_t co_pcia429_get_mode(CO_PCIA429_DEV_t* dev)
```

co_pcia429_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_pcia429_get_brdtime(CO_PCIA429_DEV_t*
dev)
```

co_pcia429_txconfig

Configuration for sender channels. You can set the speed to high (100kBaud) or to low (12.5kBaud). And the parity to odd, even or you can disable automatic parity generation (in this case the highest bit will be used as data).

```
/**
 * sets the config for a given tx channel
 *
 * @param dev device file handler
 * @param channel channel to be configured
 * @param baudrate 0 for high speed (100kBaud); 1 for low
speed (12.5kBaud)
 * @param parity 0 for dont care; 1 for even; 2 for odd
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int co_pcia429_txconfig(CO_PCIA429_DEV_t* dev, unsigned char
channel, unsigned char baudrate, unsigned char parity)
```

co_pcia429_rxconfig

Configuration for receiver channels. You can set the speed to high (100kBaud) or to low (12.5kBaud). And you can enable (odd or even) or disable the parity check.

```
/**
 * sets the config for a given rx channel
 *
 * @param dev device file handler
 * @param channel channel to be configured
 * @param baudrate 0 for high speed (100kBaud); 1 for low
speed (12.5kBaud)
 * @param parity 0 for dont care; 1 for even; 2 for odd
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
```

```
int co_pcia429_rxconfig(CO_PCIA429_DEV_t* dev, unsigned char
channel, unsigned char baudrate, unsigned char parity)
```

co_pcia429_write

Writes a 32 bit word into the sender FIFO of a given channel.

```
/**
 * writes a message into a given tx channel
 *
 * @param dev device file handler
 * @param channel channel to be configured
 * @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_pcia429_write(CO_PCIA429_DEV_t* dev, unsigned char
channel, CO_PCIA429_MESSAGE_t* message)
```

co_pcia429_read

Reads a 32 bit word 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 message data member of the struct will be the a429
word, time will be the time stamp
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int co_pcia429_read(CO_PCIA429_DEV_t* dev, unsigned char
channel, CO_PCIA429_MESSAGE_t* message)
```

co_pcia429_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 a429
word, time will be the time stamp
 * @return -EINVAL for invalid parameter or timestamp in ms
 */
double co_pcia429_timestamp_to_ms(CO_PCIA429_MESSAGE_t*
message)
```

co_pcia429_enable_loopback

Enables the loop back mode for a channel pair. Chanel pairs are given with the channels with the same numbering (eg. TX channel 0 and RX channel 0 are a pair)

```
/**
 * set a given tx/rx channel pair in loopback mode
 *
 * @param dev device file handler
 * @param channel channel to be set in loopback mode (both
tx/rx are set)
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int      co_pcia429_enable_loopback(CO_PCIA429_DEV_t*      dev,
unsigned char channel)
```

co_pcia429_disable_loopback

Disables the loopback mode for a given channel pair.

```
/**
 * reset a given tx/rx channel pair from loopback mode
 *
 * @param dev device file handler
 * @param channel channel to be reset from loopback mode
(both tx/rx are set)
 * @return -EINVAL for invalid parameter or the return value
of ioctl
 */
int      co_pcia429_disable_loopback(CO_PCIA429_DEV_t*      dev,
unsigned char channel)
```

co_pcia429_is_loopback_enabled

Returns if a given channel pair is in loopback mode.

```
/**
 * returns the loopback mode of a given tx/rx channel pair
 *
 * @param dev device file handler
 * @return -1 for error
 */
CO_PCIA429_LOOPBACK_t           co_pcia429_is_loopback_enabled
(CO_PCIA429_DEV_t* dev, unsigned char channel)
```

co_pcia429_word_2_label

Separates the label field from a ARINC429 word.

```
/**
 * filter label out of the 32 bit ARINC429 word
 *
 * @param word 32 bit ARINC429 word
```

```
 * @return label field
 */
unsigned long co_pcia429_word_2_label(unsigned long word)
```

co_pcia429_word_2_sdi

Separates the SDI field from a ARINC429 word.

```
/**
 * filter sdi out of the 32 bit ARINC429 word
 *
 * @param word 32 bit ARINC429 word
 * @return SDI field
 */
unsigned long co_pcia429_word_2_sdi(unsigned long word)
```

co_pcia429_word_2_data

Separates the data field from a ARINC429 word.

```
/**
 * filter data out of the 32 bit ARINC429 word
 *
 * @param word 32 bit ARINC429 word
 * @return data field
 */
unsigned long co_pcia429_word_2_data(unsigned long word)
```

co_pcia429_word_2_ssm

Separates the SSM field from a ARINC429 word.

```
/**
 * filter ssm out of the 32 bit ARINC429 word
 *
 * @param word 32 bit ARINC429 word
 * @return SSM field
 */
unsigned long co_pcia429_word_2_ssm(unsigned long word)
```

co_pcia429_word_2_parity

Separates the parity field from a ARINC429 word.

```
/**
 * filter parity out of the 32 bit ARINC429 word
 *
 * @param word 32 bit ARINC429 word
 * @return parity field
 */
unsigned long co_pcia429_word_2_parity(unsigned long word)
```

co_pcia429_word_builder

Builds a ARINC429 word form single fields.

```
/**
 * build a 32 bit ARINC429 word out of single fields
 * @param label       8 bit a429 label
 * @param sdi         2 bit sdi field
 * @param data       19 bit data field
 * @param ssm         2 bit ssm field
 * @param parity    1 bit parity field (according to the
channel configuration, this might be overridden!)
 * @return 32 bit ARINC429 word
 */
unsigned long co_pcia429_word_builder(unsigned long label,
unsigned long sdi, unsigned long data, unsigned long ssm,
unsigned int parity)
```

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_pcia429_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_a429load.sh script from the shell. The script creates a node under /dev. This node should be named co_pcia429-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_pcia429-0 and co_pcia429-1.

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_pcia429-0) and sends some random data on channel 0 and tries to catch the same data on rx channel 0. The example assumes that tx channel 0 is connected to rx channel 0 externally.

Chapter 4

Attachment

Related documents

hardware_reference_CO-PCIA429_v0.4 (December 2009)

History

0.1: Initial relase

0.2: added loopback

1.0: add timestamp conversion routine

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