
TPMC310-SW-95

QNX6 - Neutrino Device Driver

Isolated 2x CAN-Bus PMC (Conduction Cooled)

Version 1.0.x

User Manual

Issue 1.0.1

June 2005

TEWS TECHNOLOGIES GmbH

Am Bahnhof 7
Phone: +49-(0)4101-4058-0
e-mail: info@tews.com

25469 Halstenbek / Germany
Fax: +49-(0)4101-4058-19
www.tews.com

TEWS TECHNOLOGIES LLC

1 E. Liberty Street, Sixth Floor
Phone: +1 (775) 686 6077
e-mail: usasales@tews.com

Reno, Nevada 89504 / USA
Fax: +1 (775) 686 6024
www.tews.com

TPMC310-SW-95

Isolated 2x CAN-Bus PMC (Conduction Cooled)

QNX6-Neutrino Device Driver

This document contains information, which is proprietary to TEWS TECHNOLOGIES GmbH. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES GmbH reserves the right to change the product described in this document at any time without notice.

TEWS TECHNOLOGIES GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2005 by TEWS TECHNOLOGIES GmbH

Issue	Description	Date
1.0.0	First Issue	January 25, 2005
1.0.1	name of header files corrected, file list changed	June 23, 2005

Table of Contents

1	INTRODUCTION.....	4
2	INSTALLATION.....	5
	2.1 Build the device driver	5
	2.2 Build the example application	5
	2.3 Start the driver process.....	5
	2.4 Receive Queue Configuration.....	6
3	DEVICE INPUT/OUTPUT FUNCTIONS	7
	3.1 open()	7
	3.2 close().....	8
	3.3 devctl()	9
	3.3.1 DCMD_TPMC310_READ(_NOWAIT)	11
	3.3.2 DCMD_TP310_WRITE	13
	3.3.3 DCMD_TP310_BITTIMING	15
	3.3.4 DCMD_TP310_SETFILTER	17
	3.3.5 DCMD_TP310_BUSON.....	19
	3.3.6 DCMD_TP310_BUSOFF	20
	3.3.7 DCMD_TP310_FLUSH.....	21
	3.3.8 DCMD_TP310_CANSTATUS.....	22
	3.3.9 DCMD_TP310_ENABLE_SELFTEST	24
	3.3.10 DCMD_TP310_DISABLE_SELFTEST	25
	3.3.11 DCMD_TP310_ENABLE_LISTENONLY.....	26
	3.3.12 DCMD_TP310_DISABLE_LISTENONLY.....	27
	3.3.13 DCMD_TP310_SETLIMIT	28
	3.3.14 DCMD_TP310_TRANSCEIVER_OPERATING	29
	3.3.15 DCMD_TP310_TRANSCEIVER_SILENT	30

1 Introduction

The TPMC310-SW-95 QNX-Neutrino device driver allows the operation of a TPMC310 - Two Channel CAN PMC on QNX-Neutrino operating systems.

The TPMC310 device driver is basically implemented as a user installable Resource Manager. The standard file (I/O) functions (open, close and devctl) provide the basic interface for opening and closing a file descriptor and for performing device I/O and control operations.

The TPMC310 device driver includes the following functions:

- Transmission and receive of Standard and Extended Identifiers
- Standard bit rates from 5 kbit up to 1 Mbit and user defined bit rates
- Message acceptance filtering
- Single-Shot transmission
- Listen only mode
- Message self reception
- Programmable error warning limit

To understand all features of this device driver, it is very important to read the functional description of the SJA1000 PeliCAN controller, which is part of the engineering kit TPMC310-EK.

2 Installation

Following files are located in the directory TPMC310-SW-95 on the distribution media:

TPMC310-SW-95-SRC.tar	Driver source archive
TPMC310-SW-95.pdf	This manual in PDF format
Release.txt	Information about the Device Driver Release

Following files are stored in the TPMC310-SW-95-SRC.tar archive:

/driver/tpmc310.c	Driver source code
/driver/tpmc310.h	Definitions and data structures for driver and application
/driver/tpmc310def.h	Device driver include
/driver/node.c	Queue management source code
/driver/node.h	Queue management definitions
/example/example.c	Example application

For installation copy the tar-archive into the /usr/src directory and unpack it (e.g. `tar -xvf TPMC310-SW-95-SRC.tar`). After that the necessary directory structure for the automatic build and the source files are available underneath the new directory called *tpmc310*.

It is absolutely important to extract the TPMC310 tar archive in the /usr/src directory. Otherwise the automatic build with make will fail.

2.1 Build the device driver

Change to the /usr/src/tpmc310/driver directory

Execute the Makefile:

```
# make install
```

After successful completion the driver binary (tpmc310) will be installed in the /bin directory.

2.2 Build the example application

Change to the /usr/src/tpmc310/example directory

Execute the Makefile:

```
# make install
```

After successful completion the example binary (*tp310exam*) will be installed in the /bin directory.

2.3 Start the driver process

To start the TPMC310 device driver respective you have to enter the process name with optional parameter from the command shell or in the startup script.

```
tpmc310 [-v] &
```

The TPMC310 Resource Manager registers created devices in the Neutrinos pathname space under following names.

```
/dev/tpmc310_0  
/dev/tpmc310_1  
...  
/dev/tpmc310_x
```

This pathname must be used in the application program to open a path to the desired TPMC310 device.

```
fd = open("/dev/tpmc310_0", O_RDWR);
```

For debugging you can start the TPMC310 Resource Manager with the `-v` option. Now the Resource Manager will print versatile information about TPMC310 configuration and command execution on the terminal window.

```
tpmc310 -v &
```

2.4 Receive Queue Configuration

Received CAN messages will be stored in a FIFO buffer. The depth of the FIFO can be adapted by changing the following symbol in `tpmc310def.h`.

`TP310_RX_FIFO_SIZE`

Defines the depth of the message FIFO buffer (default = 100). Valid numbers are in range between 1 and MAXINT.

3 Device Input/Output functions

This chapter describes the interface to the device driver I/O system.

3.1 open()

NAME

open() - open a file descriptor

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

```
int open (const char *pathname, int flags)
```

DESCRIPTION

The *open* function creates and returns a new file descriptor for the TPMC310 named by pathname. The flags argument controls how the file is to be opened. TPMC310 devices must be opened O_RDWR.

EXAMPLE

```
int fd;

fd = open("/dev/tpmc310_0", O_RDWR);
```

RETURNS

The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of -1 is returned. The global variable errno contains the detailed error code.

ERRORS

Returns only Neutrino specific error codes, see Neutrino Library Reference.

SEE ALSO

Library Reference - open()

3.2 close()

NAME

close() – close a file descriptor

SYNOPSIS

```
#include <unistd.h>
```

```
int close (int filedes)
```

DESCRIPTION

The close function closes the file descriptor *filedes*.

EXAMPLE

```
int fd;

...

if (close(fd) != 0)
{
    /* handle close error conditions */
}
```

RETURNS

The normal return value from close is 0. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

Returns only Neutrino specific error code, see Neutrino Library Reference.

SEE ALSO

Library Reference - close()

3.3 devctl()

NAME

devctl() – device control functions

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>
#include <devctl.h>
```

```
int devctl
(
    int          filedes,
    int          dcmd,
    void         *data_ptr,
    size_t       n_bytes,
    int          *dev_info_ptr
)
```

DESCRIPTION

The devctl function sends a control code directly to a device, specified by *filedes*, causing the corresponding device to perform the requested operation.

The argument *dcmd* specifies the control code for the operation.

The arguments *data_ptr* and *n_bytes* depends on the command and will be described for each command in detail later in this chapter. Usually *data_ptr* points to a buffer that passes data between the user task and the driver and *n_bytes* defines the size of this buffer.

The argument *dev_info_ptr* is unused for the TPMC310 driver and should be set to NULL.

The following devctl command codes are defined in *tpmc310.h*:

Value	Description
<i>DCMD_TP310_READ</i>	Read a CAN message
<i>DCMD_TP310_READ_NOWAIT</i>	Read a CAN message and return immediately if queue is empty
<i>DCMD_TP310_WRITE</i>	Write message to the CAN bus
<i>DCMD_TP310_BITTIMING</i>	Setup a new bit timing
<i>DCMD_TP310_SETFILTER</i>	Setup acceptance filter
<i>DCMD_TP310_BUSON</i>	Enter the bus on state
<i>DCMD_TP310_BUSOFF</i>	Enter the bus off state
<i>DCMD_TP310_FLUSH</i>	Flush one or all receive queues
<i>DCMD_TP310_CANSTATUS</i>	Returns CAN controller status information
<i>DCMD_TP310_ENABLE_SELFTEST</i>	Enable self test mode

<i>DCMD_TP310_DISABLE_SELFTEST</i>	Disable self test mode
<i>DCMD_TP310_ENABLE_LISTENONLY</i>	Enable listen only mode
<i>DCMD_TP310_DISABLE_LISTENONLY</i>	Disable listen only mode
<i>DCMD_TP310_SETLIMIT</i>	Set new error warning limit

See behind for more detailed information on each control code.

To use these TPMC310 specific control codes the header file `tpmc310.h` must be included in the application.

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in `errno!`).

ERRORS

Returns only Neutrino specific error codes, see Neutrino Library Reference.

Other function dependent error codes will be described for each `devctl` code separately. Note, the TPMC310 driver always returns standard QNX error codes.

SEE ALSO

Library Reference - `devctl()`

3.3.1 DCMD_TPMC310_READ(_NOWAIT)

NAME

DCMD_TPMC310_READ(_NOWAIT) – Read a CAN message

DESCRIPTION

The read function reads a CAN message from the driver receive queue. A pointer to the callers message buffer (TP310_MSG_BUF) and the size of this structure are passed by the parameters *data_ptr* and *n_bytes* to the device.

The TP310_MSG_BUF structure has the following layout:

```
typedef struct {
    unsigned long      Identifier;
    unsigned char      IOFlags;
    unsigned char      MsgLen;
    unsigned char      Data[8];
    long               Timeout;
    unsigned char      Status;
} TP310_MSG_BUF, *PTP310_MSG_BUF;
```

unsigned long Identifier

Receives the message identifier of the read CAN message.

unsigned char IOFlags

Receives CAN message attributes as a set of bit flags. The following attribute flags are possible:

TP310_EXTENDED	Set if the received message is an extended message frame. Reset for standard message frames.
TP310_REMOTE_FRAME	Set if the received message is a remote transmission request (RTR) frame.

unsigned char MsgLen

Receives the number of message data bytes (0...8).

unsigned char Data[8]

This buffer receives up to 8 data bytes. *Data[0]* receives message Data 0, *Data[1]* receives message Data 1 and so on.

long Timeout

Specifies the amount of time (in seconds) the caller is willing to wait for execution of read. A value of 0 means wait indefinitely.

unsigned char Status

Receives status information about overrun conditions either in the CAN controller or intermediate software FIFO.

TP310_SUCCESS	No messages lost
TP310_FIFO_OVERRUN	One or more messages was overwritten in the receive queue FIFO. This problem occurs if the FIFO is too small for the application read interval.
TP310_MSGOBJ_OVERRUN	One or more messages were overwritten in the CAN controller message FIFO because the interrupt latency is too large. Reduce the CAN bit rate or upgrade the system speed.

EXAMPLE

```
int fd;
int result;
TP310_MSG_BUF MsgBuf;

MsgBuf.Timeout = 10;    /* seconds */

result = devctl(    fd,
                  DCMD_TP310_READ,
                  &MsgBuf,
                  sizeof(MsgBuf),
                  NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

<i>EINVAL</i>	Invalid argument. This error code is returned if the size of the message buffer is too small.
<i>ENOMEM</i>	No memory available to allocated resources to handle the read command.
<i>ECONNREFUSED</i>	The controller is in bus OFF state and no message is available in the specified receive queue. Note, as long as CAN messages are available in the receive queue FIFO, bus OFF conditions were not reported by a read function. This means you can read all CAN messages out of the receive queue FIFO during bus OFF state without an error result.
<i>ENODATA</i>	Currently no CAN message available for read (only DCMD_TP310_READ_NOWAIT).
<i>ETIMEDOUT</i>	The allowed time to finish the read request is elapsed.

SEE ALSO

Library Reference - devctl()

3.3.2 DCMD_TP310_WRITE

NAME

DCMD_TP310_WRITE - Write a CAN message

DESCRIPTION

This devctl function writes a message to the CAN bus. A pointer to the callers message buffer (*TP310_MSG_BUF*) and the size of this structure are passed by the parameters *data_ptr* and *n_bytes* to the device.

If the CAN controller is busy transmitting a message the caller becomes blocked until all previous pending requests are serviced or a timeout occurs.

The TP310_MSG_BUF structure has the following layout:

```
typedef struct {
    unsigned long    Identifier;
    unsigned char   IOFlags;
    unsigned char   MsgLen;
    unsigned char   Data[8];
    long            Timeout;
    unsigned char   Status;
} TP310_MSG_BUF, *PTP310_MSG_BUF;
```

unsigned long Identifier

Contains the message identifier of the CAN message to write.

unsigned char IOFlags

Contains a set of bit flags, which define message attributes and controls the write operation. To set more than one bit flag the predefined macros must be binary OR'ed.

TP310_EXTENDED	Transmit an extended message frame. If this macro isn't set or the "dummy" macro TP310_STANDARD is set a standard frame will be transmitted.
TP310_REMOTE_FRAME	A remote transmission request (RTR bit is set) will be transmitted.
TP310_SINGLE_SHOT	No re-transmission will be performed if an error occurred or the arbitration will be lost during transmission (single-shot transmission).
TP310_SELF_RECEPTION	The message will be transmitted and simultaneously received if the acceptance filter is set to the corresponding identifier.

unsigned char MsgLen

Contains the number of message data bytes (0...8).

unsigned char Data[8]

This buffer contains up to 8 data bytes. Data[0] contains message Data 0, Data[1] contains message Data 1 and so on.

long Timeout

Specifies the amount of time (in seconds) the caller is willing to wait for execution of write.

unsigned char Status

Unused for this control function. Can be 0.

EXAMPLE

```
int fd;
int result;
TP310_MSG_BUF MsgBuf;

MsgBuf.Identifier = 1234;
MsgBuf.Timeout = 2; /* sec*/
MsgBuf.IOFlags = TP310_EXTENDED;
MsgBuf.MsgLen = 2;
MsgBuf.Data[0] = 0xaa;
MsgBuf.Data[1] = 0x55;

result = devctl( fd,
                 DCMD_TP310_WRITE,
                 &MsgBuf,
                 sizeof(MsgBuf),
                 NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

<i>EINVAL</i>	Invalid argument. This error code is returned if the size of the message buffer is too small.
<i>ENOMEM</i>	No memory available to allocated resources to handle the read command.
<i>ECONNREFUSED</i>	The controller is in bus OFF state and unable to transmit messages.
<i>EMSGSIZE</i>	Invalid message size. <i>MsgBuf.MsgLen</i> must be in range between 0 and 8.
<i>ETIMEDOUT</i>	The allowed time to finish the write request is elapsed.

SEE ALSO

Library Reference - devctl()

3.3.3 DCMD_TP310_BITTIMING

NAME

DCMD_TP310_BITTIMING – Setup new bit timing

DESCRIPTION

This devctl function modifies the bit timing register of the CAN controller to setup a new CAN bus transfer speed. A pointer to the callers parameter buffer (*TP310_TIMING*) and the size of this structure are passed by the parameters *data_ptr* and *n_bytes* to the device.

Keep in mind to setup a valid bit timing value before changing into the Bus On state.

The *TP310_TIMING* structure has the following layout:

```
typedef struct {
    unsigned short    TimingValue;
    unsigned short    ThreeSamples;
}TP310_TIMING, *PTP310_TIMING;
```

unsigned short TimingValue

This parameter holds the new value for the bit timing register 0 (bit 0...7) and for the bit timing register 1 (bit 8...15). Possible transfer rates are between 5 Kbit per second and 1 Mbit per second. The include file 'tpmc310.h' contains predefined transfer rate symbols (TP310_5KBIT ... TP310_1MBIT).

For other transfer rates please follow the instructions of the *SJA1000 Product Specification*, which is also part of the engineering kit TPMC310-EK.

unsigned short ThreeSamples

If this parameter is TRUE (1) the CAN bus is sampled three times per bit time instead of one.

Use one sample point for faster bit rates and three sample points for slower bit rate to make the CAN bus more immune against noise spikes.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;
TP310_TIMING BitTimingParam;

BitTimingParam.TimingValue = TP310_100KBIT;
BitTimingParam.ThreeSamples = FALSE;

result = devctl(    fd,
                   DCMD_TP310_BITTIMING,
                   &BitTimingParam,
                   sizeof(BitTimingParam),
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

EACCES Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state before changing the bit timing.

SEE ALSO

Library Reference - devctl().

tpmc310.h for predefined bus timing constants.

SJA1000 Product Specification Manual – 6.5.1/2 BUS TIMING REGISTER.

3.3.4 DCMD_TP310_SETFILTER

NAME

DCMD_TP310_SETFILTER - Setup acceptance filter

DESCRIPTION

This devctl function modifies the acceptance filter of the specified CAN controller device.

The acceptance filter compares the received identifier with the acceptance filter and decides whether a message should be accepted or not. If a message passes the acceptance filter it is stored in the receive FIFO.

The acceptance filter is defined by the acceptance code registers and the acceptance mask registers. The bit patterns of messages to be received are defined in the acceptance code register.

The corresponding acceptance mask registers allow defining certain bit positions to be "don't care" (a 1 at a bit position means "don't care").

A pointer to the callers parameter buffer (*TP310_FILTER*) and the size of this structure are passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP310_FILTER* structure has the following layout:

```
typedef struct {
    int                SingleFilter;
    unsigned long      AcceptanceCode;
    unsigned long      AcceptanceMask;
}TP310_FILTER, *PTP310_FILTER;
```

int SingleFilter

Set TRUE (1) for single filter mode.
Set FALSE (0) for dual filter mode.

unsigned long AcceptanceCode

The contents of this parameter will be written to acceptance code register of the controller.

unsigned long AcceptanceMask

The contents of this parameter will be written to the acceptance mask register of the controller.

A detailed description of the acceptance filter and possible filter modes can be found in the SJA1000 Product Specification Manual.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;
TP310_FILTER AcceptFilter;

/* Not relevant because all bits are "don't care" */
AcceptFilter.AcceptanceCode = 0x0;

/* Mark all bit position don't care */
AcceptFilter.AcceptanceMask = 0xffffffff;

/* Single Filter Mode */
AcceptFilter.SingleFilter = 1;    // TRUE

result = devctl(    fd,
                   DCMD_TP310_SETFILTER,
                   & AcceptFilter,
                   sizeof(AcceptFilter),
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

<i>EACCES</i>	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state before changing the acceptance filter.
---------------	---

SEE ALSO

Library Reference - `devctl()`.

SJA1000 Product Specification Manual – 6.4.15 ACCEPTANCE FILTER

3.3.5 DCMD_TP310_BUSON

NAME

DCMD_TP310_BUSON - Enter the bus ON state

DESCRIPTION

This devctl function sets the specified CAN controller into the bus ON state.

After an abnormal rate of occurrences of errors on the CAN bus or after driver startup, the CAN controller enters the Bus OFF state. This control function resets the "reset mode" bit in the mode register. The CAN controller begins the bus OFF recovery sequence and resets the transmit and receive error counters. If the CAN controller counts 128 packets of 11 consecutive recessive bits on the CAN bus, the Bus Off state is exited.

Before the driver is able to communicate over the CAN bus after driver startup, this control function must be executed.

EXAMPLE

```
int fd;
int result;

result = devctl(    fd,
                   DCMD_TP310_BUSON,
                   NULL,
                   0,
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

ECONNREFUSED Unable to enter the Bus ON mode.

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*.

3.3.7 DCMD_TP310_FLUSH

NAME

DCMD_TP310_FLUSH - Flush the received message FIFO

DESCRIPTION

This devctl function flushes the FIFO buffer of received CAN messages.

EXAMPLE

```
int fd;
int result;

result = devctl(    fd,
                   DCMD_TP310_FLUSH,
                   NULL,
                   0,
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

No function specific errors.

SEE ALSO

Library Reference - devctl().

3.3.8 DCMD_TP310_CANSTATUS

NAME

DCMD_TP310_CANSTATUS - Returns CAN controller status information

DESCRIPTION

This devctl function returns the actual contents of several CAN controller registers for diagnostic purposes.

A pointer to the callers status buffer (*TP310_STATUS*) and the size of this structure are passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP310_STATUS* structure has the following layout:

```
typedef struct {
    unsigned char    ArbitrationLostCapture;
    unsigned char    ErrorCodeCapture;
    unsigned char    TxErrorCounter;
    unsigned char    RxErrorCounter;
    unsigned char    ErrorWarningLimit;
    unsigned char    StatusRegister;
    unsigned char    ModeRegister;
    unsigned char    RxMessageCounterMax;
} TP310_STATUS, *PTP310_STATUS;
```

unsigned char ArbitrationLostCapture

Contents of the arbitration lost capture register. This register contains information about the bit position of losing arbitration.

unsigned char ErrorCodeCapture

Contents of the error code capture register. This register contains information about the type and location of errors on the bus.

unsigned char TxErrorCounter

Contents of the TX error counter register. This register contains the current value of the transmit error counter.

unsigned char RxErrorCounter

Contents of the TX error counter register. This register contains the current value of the receive error counter.

unsigned char ErrorWarningLimit

Contents of the error warning limit register.

unsigned char StatusRegister

Contents of the status register.

unsigned char ModeRegister

Contents of the mode register.

unsigned char RxMessageCounterMax

Contains the peak value of messages in the receive FIFO. This internal counter value will be reset to 0 after reading.

EXAMPLE

```
int fd;
int result;
TP310_STATUS CanStatus;

result = devctl(    fd,
                   DCMD_TP310_CANSTATUS,
                   &CanStatus,
                   sizeof(CanStatus),
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

No function specific errors.

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual

3.3.9 DCMD_TP310_ENABLE_SELFTEST

NAME

DCMD_TP310_ENABLE_SELFTEST - Enable self test mode

DESCRIPTION

This devctl function enables the self test facility of the SJA1000 CAN controller.

In this mode a full node test is possible without any other active node on the bus using the self reception facility. The CAN controller will perform a successful transmission even if there is no acknowledge received.

Also in self test mode the normal functionality is given, that means the CAN controller is able to receive messages from other nodes and can transmit message to other nodes if any connected.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;

result = devctl(    fd,
                   DCMD_TP310_ENABLE_SELFTEST,
                   NULL,
                   0,
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

EACCES

The CAN controller is in operating mode. This mode can be changed only in reset mode.

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*

3.3.10 DCMD_TP310_DISABLE_SELFTEST

NAME

DCMD_TP310_DISABLE_SELFTEST - Disable self test mode

DESCRIPTION

This devctl function disables the self test facility of the SJA1000 CAN controller, which was before enabled with the devctl command DCMD_TP310_ENABLE_SELFTEST.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;

result = devctl(    fd,
                   DCMD_TP310_DISABLE_SELFTEST,
                   NULL,
                   0,
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

EACCES The CAN controller is in operating mode. This mode can be changed only in reset mode.

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*

3.3.11 DCMD_TP310_ENABLE_LISTENONLY

NAME

DCMD_TP310_ENABLE_LISTENONLY - Enable listen only mode

DESCRIPTION

This devctl function enables the listen only facility of the SJA1000 CAN controller.

In this mode the CAN controller would give no acknowledge to the CAN-bus, even if a message is received successfully. Message transmission is not possible. All other functions can be used like in normal mode.

This mode can be used for software driver bit rate detection and 'hot-plugging'.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;

result = devctl(    fd,
                   DCMD_TP310_ENABLE_LISTENONLY,
                   NULL,
                   0,
                   NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

<i>EACCES</i>	The CAN controller is in operating mode. This mode can be changed only in reset mode.
---------------	---

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*

3.3.12 DCMD_TP310_DISABLE_LISTENONLY

NAME

DCMD_TP310_DISABLE_LISTENONLY - Disable listen only mode

DESCRIPTION

This devctl function disables the listen only facility of the SJA1000 CAN controller, which was before enabled with the devctl command DCMD_TP310_ENABLE_LISTENONLY.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;

result = devctl(    fd,
                  DCMD_TP310_DISABLE_LISTENONLY,
                  NULL,
                  0,
                  NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

<i>EACCES</i>	The CAN controller is in operating mode. This mode can be changed only in reset mode.
---------------	---

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*

3.3.13 DCMD_TP310_SETLIMIT

NAME

DCMD_TP310_SETLIMIT - Disable listen only mode

DESCRIPTION

This devctl function sets a new error warning limit in the corresponding CAN controller register. The default value (after hardware reset) is 96.

The new error warning limit will be set in an unsigned char variable. A pointer to this variable is passed by the parameters *data_ptr* to the driver. The size of this variable is passed by the parameter *n_bytes* to the driver.

This devctl command will be accepted only in reset mode (BUSOFF). Enter DCMD_TP310_BUSOFF first otherwise you will get an error (EACCES).

EXAMPLE

```
int fd;
int result;
unsigned char ErrorLimit

ErrorLimit = 20;

result = devctl(    fd,
                  DCMD_TP310_SETLIMIT,
                  &ErrorLimit,
                  sizeof(ErrorLimit),
                  NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

<i>EACCES</i>	The CAN controller is in operating mode. This mode can be changed only in reset mode.
---------------	---

SEE ALSO

Library Reference - devctl().

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*

3.3.14 DCMD_TP310_TRANSCEIVER_OPERATING

NAME

DCMD_TP310_TRANSCEIVER_OPERATING – Switch transceiver into Operating Mode

DESCRIPTION

This devctl function switches the onboard transceivers for the specific channel into Operating Mode.

EXAMPLE

```
int fd;
int result;

result = devctl( fd,
                DCMD_TP310_TRANSCEIVER_OPERATING,
                NULL,
                0,
                NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

none

SEE ALSO

Library Reference - devctl().

TPMC310 Hardware User Manual

SJA1000 Product Specification Manual – 6.4.3 MODE REGISTER (MOD)

3.3.15 DCMD_TP310_TRANSCEIVER_SILENT

NAME

DCMD_TP310_TRANSCEIVER_SILENT – Switch transceiver into Silent Mode

DESCRIPTION

This devctl function switches the onboard transceivers for the specific channel into Silent Mode.

EXAMPLE

```
int fd;
int result;

result = devctl( fd,
                 DCMD_TP310_TRANSCEIVER_SILENT,
                 NULL,
                 0,
                 NULL);

if (result != EOK) {
    /* process devctl() error */
}
```

ERRORS

none

SEE ALSO

Library Reference - devctl().

TPMC310 Hardware User Manual

SJA1000 Product Specification Manual – 6.4.3 *MODE REGISTER (MOD)*