The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV010-SW-42

VxWorks Device Driver

Isolated 2x CAN Bus

Version 4.2.x

User Manual

Issue 4.2.0

March 2022

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7   25469 Halstenbek, Germany
Phone: +49 (0) 4101 4058 0   Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com   www.tews.com

TDRV010-SW-42

VxWorks Device Driver

Isolated 2x CAN Bus

Supported Modules:
TPMC310
TPMC810

Issue	Description	Date
1.0.0	First Issue	October 12, 2005
1.0.1	Review	November 8, 2005
1.1.0	Updated CAN Controller Speed Range and General Review	May 30, 2006
1.1.1	Revision	November 1, 2006
2.0.0	Support for VxBus and API description added	November 16, 2009
2.0.1	Legacy vs. VxBus Driver modified	March 26, 2010
2.0.2	tdrv010Init() function added	September 22, 2010
3.0.0	API revised and documented, Basic I/O functions removed	October 23, 2012
3.0.1	General Revision, Related Documents modified	August 25, 2015
4.0.0	VxWorks 7 Support added Installation moved into separate manual	February 17, 2017
4.1.0	VxWorks 7R2 Support added tdrv010GetModuleInfo() example corrected	April 30, 2019
4.2.0	ioctl for RTP-Support modified	March 24, 2022

Table of Contents

1 Introduction

The TDRV010-SW-42 VxWorks device driver software allows the operation of the supported modules conforming to the VxWorks I/O system specification.

The TDRV010-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new VxBus-enabled driver model. The VxBus-enabled driver is recommended for new developments with later VxWorks 6.x release and mandatory for VxWorks SMP systems.

The driver provides an application programming interface (API) which allows OS independent access to the devices for compatibility between different OS versions and OS.

To prevent the application program for losing data, incoming messages will be stored in a message FIFO with a depth of 100 messages.

The TDRV010-SW-42 device driver supports the following features:

- Transmit and receive Standard and Extended Identifiers
- Standard bit rates from 50 kbit/s up to 1.0 Mbit/s and user defined bit rates
- Message acceptance filtering
- Single-Shot transmission
- Listen only mode
- Message self-reception
- Programmable error warning limit

The TDRV010-SW-42 supports the modules listed below:

TPMC310	Isolated 2x CAN	(PMC, Conduction Cooled, Silent Mode Options)
TPMC810	Isolated 2x CAN	(PMC)

In this document all supported modules and devices will be called TDRV010. Specials for certain devices will be advised.

To get more information about the features and use of the supported devices it is recommended to read the manuals listed below.

TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide
TPMC310 and TPMC810 User Manual
SJA1000 CAN Controller Manual

2 API Documentation

2.1 General Functions

2.1.1 tdrv010Open

NAME

tdrv010Open – opens a device.

SYNOPSIS

```
TDRV010_HANDLE tdrv010Open
(
        char            *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle must be opened by a call to this function. If the legacy TDRV010 driver is used, this function will also install the legacy driver and create devices with the first call. The VxBus TDRV010 driver will be installed automatically by the VxBus system.

The tdrv010Open function can be called multiple times (e.g. in different tasks)

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first CAN channel on the first TDRV010 device is named “/tdrv010/0/0”, the second channel is named “/tdrv010/0/1”. The first CAN channel on the second TDRV010 device is named “/tdrv010/1/0” and so on.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tdrv010Open(“/tdrv010/0/0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. An error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno.*

The error code is a standard error code set by the I/O system.

2.1.2 tdrv010Close

NAME

tdrv010Close – closes a device.

SYNOPSIS

```
TDRV010_STATUS tdrv010Close
(
    TDRV010_HANDLE        hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

/*
** close the device
*/
result = tdrv010Close(hdl);
if (result != TDRV010_OK)
{
    /* handle close error */
}
```

RETURNS

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.3 tdrv010GetModuleInfo

NAME

tdrv010GetModuleInfo – get information of the module

SYNOPSIS

```
TDRV010_STATUS tdrv010GetModuleInfo
(
        TDRV010_HANDLE              hdl,
        unsigned int                *pModuleType,
        unsigned int                *pChannelNo,
        TDRV010_PCIINFO_BUF         *pPciInfoBuf
)
```

DESCRIPTION

This function returns information about the module, including module type, local channel number and PCI header as well as the PCI localization.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

pModuleType

This argument is a pointer to an *unsigned int* (32bit) data buffer, where the module type is returned. Possible values are:

Value	Description
TDRV010_MODTYPE_TPMC310	Current module is a TPMC310
TDRV010_MODTYPE_TPMC810	Current module is a TPMC810

pChannelNo

This argument is a pointer to an *unsigned int* (32bit) data buffer, where the local channel number of the device is returned. Possible values are 0 or 1.

pPciInfoBuf

This argument is a pointer to the structure TDRV010_PCIINFO_BUF that receives information of the module PCI header.

```
typedef struct
{
        unsigned short      vendorId;
        unsigned short      deviceId;
        unsigned short      subSystemId;
        unsigned short      subSystemVendorId;
        int                 pciBusNo;
        int                 pciDevNo;
        int                 pciFuncNo;
} TDRV010_PCIINFO_BUF;
```

vendorId

PCI module vendor ID.

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module sub system vendor ID

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE          hdl;
TDRV010_STATUS          result;
TDRV010_PCIINFO_BUF     pciInfoBuf;
unsigned int            moduleType;
unsigned int            channelNo

/*
** get module PCI information
*/
result = tdrv010GetModuleInfo( hdl, &moduleType, &channelNo, &pciInfoBuf );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV010_ERR_INVAL	Specified pointer is invalid.

2.1.4 tdrv010GetControllerStatus

NAME

tdrv010GetControllerStatus – Get CAN controller status information

SYNOPSIS

```
TDRV010_STATUS tdrv010GetControllerStatus
(
        TDRV010_HANDLE              hdl,
        TDRV010_STATUS_BUF          *pCANStatus
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

pCANStatus

This parameter points to a TDRV010_STATUS_BUF buffer, which receives the CAN controller status:

```
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;
        unsigned char       PLDControl;
} TDRV010_STATUS_BUF;
```

ArbitrationLostCapture

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

ErrorCodeCapture

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

TxErrorCounter

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

RxErrorCounter

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

ErrorWarningLimit

Contents of the error warning limit register.

StatusRegister

Contents of the status register.

ModeRegister

Contents of the mode register.

RxMessageCounterMax

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

PLDControl

If available this parameter retrieves the content of the PLD Control Register. For non TPMC310 modules this parameter retrieves a value greater or equal 0x80 (means invalid). On TPMC310 devices the retrieved value will describe exactly the content of PLDControlReg[5:0].

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE          hdl;
TDRV010_STATUS          result;
TDRV010_STATUS_BUF      CanStatus;

result = tdrv010GetControllerStatus( hdl, &CanStatus );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.

SEE ALSO

SJA1000 Product Specification Manual

2.2 Communication Functions

2.2.1 tdrv010Read

NAME

tdrv010Read – Read a CAN message

SYNOPSIS

```
TDRV010_STATUS tdrv010Read
(
      TDRV010_HANDLE          hdl,
      int                     Timeout,
      unsigned int            *pIdentifier,
      unsigned char           *pIOFlags,
      unsigned char           *pStatus,
      int                     *pLength,
      unsigned char           *pData
)
```

DESCRIPTION

This function reads a CAN message from the device driver receive queue. If no data is available, the function blocks until data is received or the specified timeout has expired.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

Timeout

This parameter specifies the maximum time (in milliseconds) the function will block and wait for data if no data is available. Specify -1 to wait indefinitely, or 0 to return immediately.

pIdentifier

This parameter is a pointer to an *unsigned int* (32bit) value where the CAN message identifier is stored.

pIOFlags

This parameter is a pointer to an *unsigned char* (8bit) value where CAN message attributes as a set of bit flags are stored. The following attribute flags are possible:

Value	**Description**
TDRV010_EXTENDED	Set if the received message is an extended message frame. Reset for standard message frames.
TDRV010_REMOTE_FRAME	Set if the received message is a remote transmission request (RTR) frame.

pStatus

This parameter is a pointer to an *unsigned char* (8bit) value where status information about overrun conditions either in the CAN controller or intermediate software FIFO is stored. The following values are possible:

Value	**Description**
TDRV010_SUCCESS	No messages lost
TDRV010_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.
TDRV010_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.

pLength

This parameter is a pointer to an *int* value where the length of the received CAN message (number of bytes) is stored. Possible values are 0..8.

pData

This parameter is a pointer to an *unsigned char* array where the received CAN message is stored. This buffer receives up to 8 data bytes. pData[0] receives message Data 0, pData[1] receives message Data 1 and so on.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;
int                Timeout;
unsigned int       Identifier;
unsigned char      IOFlags;
unsigned char      Status;
int                Length;
unsigned char      Data[8];
```

…

```
…

/*
** Read a CAN message from the device.
** If no data is available, wait 5000ms for incoming messages.
*/
Timeout = 5000;
result = tdrv010Read(   hdl,
                        Timeout,
                        &Identifier,
                        &IOFlags,
                        &Status,
                        &Length,
                        &Data[0] );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_TIMEOUT	Read was blocked and the allowed time has elapsed.
TDRV010_ERR_BUSOFF	The controller is in bus OFF state and no message is available in the receive queue. Note, as long as CAN messages are available in the receive queue FIFO, bus OFF conditions were not reported by the read function. This means you can read all CAN messages out of the receive queue FIFO during bus OFF state without an error result.

2.2.2 tdrv010Write

NAME

tdrv010Write – Write a CAN message

SYNOPSIS

```
TDRV010_STATUS tdrv010Write
(
        TDRV010_HANDLE          hdl,
        int                     Timeout,
        unsigned int            Identifier,
        unsigned char           IOFlags,
        int                     Length,
        unsigned char           *pData
)
```

DESCRIPTION

This function writes a CAN message to the CAN bus. The function waits for the message to be sent until the specified timeout has expired.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

Timeout

Specifies the amount of time (in milliseconds) the caller is willing to wait for execution of write request. A value of -1 means wait indefinitely. If Timeout is set to 0 the function will return immediately after initiating the write in the CAN controller.

Identifier

Contains the message identifier of the CAN message to write.

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.

Value	**Description**
TDRV010_EXTENDED	Transmit an extended message frame. If this macro isn't set or the "dummy" macro TDRV010_STANDARD is set a standard frame will be transmitted.
TDRV010_REMOTE_FRAME	A remote transmission request (RTR bit is set) will be transmitted.
TDRV010_SINGLE_SHOT	No re-transmission will be performed if an error occurred or the arbitration will be lost during transmission (single-shot transmission).
TDRV010_SELF_RECEPTION	The message will be transmitted and simultaneously received if the acceptance filter is set to the corresponding identifier.

Length

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

pData

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

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;
int                Timeout;
unsigned int       Identifier;
unsigned char      IOFlags;
int                Length;
unsigned char      Data[8];

/*
** Write an extended CAN message to the device.
*/
Identifier    = 1234;
Timeout       = 5000;
IOFlags       = TDRV010_EXTENDED | TDRV010_SINGLE_SHOT;
MsgLen        = 2;
Data[0]       = 0xaa;
Data[1]       = 0x55;
…
```

```
…

result = tdrv010Write(  hdl,
                        Timeout,
                        Identifier,
                        IOFlags,
                        Length,
                        &Data[0] );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_TIMEOUT	The allowed time to finish the write request is elapsed.
TDRV010_ERR_BUSOFF	The controller is in bus OFF state and unable to transmit messages.
TDRV010_ERR_INVAL	Illegal message length (valid range is 0...8).
TDRV010_ERR_COMM	Transmission failed in single shot mode.

2.3 Configuration Functions

2.3.1 tdrv010SetFilter

NAME

tdrv010SetFilter – Configure Acceptance Filter

SYNOPSIS

```
TDRV010_STATUS tdrv010SetFilter
(
        TDRV010_HANDLE              hdl,
        int                         SingleFilter,
        unsigned int                AcceptanceCode,
        unsigned int                AcceptanceMask
)
```

DESCRIPTION

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

This function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

SingleFilter

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

AcceptanceCode

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

AcceptanceMask

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

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;
int                SingleFilter;
unsigned int       AcceptanceCode;
unsigned int       AcceptanceMask;

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

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

/* Single Filter Mode */
SingleFilter = 1;

result = tdrv010SetFilter(   hdl,
                             SingleFilter,
                             AcceptanceCode,
                             AcceptanceMask );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state before changing the acceptance filter.

SEE ALSO

tdrv010exa.c for a programming example.

SJA1000 Product Specification Manual – *6.4.15 ACCEPTANCE FILTER*

2.3.2 tdrv010SetBitTiming

NAME

tdrv010SetBitTiming – Modify CAN Bus transfer speed

SYNOPSIS

```
TDRV010_STATUS tdrv010SetBitTiming
(
        TDRV010_HANDLE                  hdl,
        unsigned short                  TimingValue,
        int                             UseThreeSamples
)
```

DESCRIPTION

This function modifies the bit timing registers of the CAN controller to setup a new CAN bus transfer speed.

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

This function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

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 50 Kbit per second and 1 Mbit per second. The include file 'tdrv010api.h' contains predefined transfer rate symbols (TDRV010_5KBIT ... TDRV010_1MBIT).
For other transfer rates please follow the instructions of the *SJA1000 Product Specification*, which is also part of the TPMC310 or TPMC810 engineering documentation.

UseThreeSamples

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

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;
int                UseThreeSamples;
unsigned short     TimingValue;

TimingValue        = TDRV010_100KBIT;
UseThreeSamples    = FALSE;

result = tdrv010SetBitTiming(    hdl,
                                 TimingValue,
                                 UseThreeSamples );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state before changing the bit timing.

SEE ALSO

tdrv010exa.c for a programming example.

tdrv010api.h for predefined bus timing constants.

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

2.3.3 tdrv010Start

NAME

tdrv010Start – Set CAN controller into BUSON state

SYNOPSIS

```
TDRV010_STATUS tdrv010Start
(
    TDRV010_HANDLE          hdl
)
```

DESCRIPTION

This function sets the specified CAN controller into the BUSON state.

After an abnormal rate of occurrences of errors on the CAN bus or after driver startup, the CAN controller enters the BUSOFF state. This control function resets the "reset mode" bit in the mode register. The CAN controller begins the bus OFF recovery sequence and resets 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.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010Start( hdl );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_BUSOFF	Unable to enter the Bus ON mode.

SEE ALSO

tdrv010exa.c for a programming example.

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

2.3.4 tdrv010Stop

NAME

tdrv010Stop – Set CAN controller into BUSOFF state

SYNOPSIS

```
TDRV010_STATUS tdrv010Stop
(
        TDRV010_HANDLE hdl
)
```

DESCRIPTION

This function sets the specified CAN controller into the bus OFF state.

After execution of this control function the CAN controller is completely removed from the CAN bus and cannot communicate until the control function tdrv010Start() is executed.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010Stop( hdl );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_IO	Unable to enter the Bus OFF mode.

SEE ALSO

tdrv010exa.c for a programming example.

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

2.3.5 tdrv010FlushReceiveFifo

NAME

tdrv010FlushReceiveFifo – Flush software receive FIFO

SYNOPSIS

```
TDRV010_STATUS tdrv010FlushReceiveFifo
(
    TDRV010_HANDLE          hdl
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010FlushReceiveFifo( hdl );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.

2.3.6 tdrv010SelftestEnable

NAME

tdrv010SelftestEnable – Enable self test facility

SYNOPSIS

```
TDRV010_STATUS tdrv010SelftestEnable
(
        TDRV010_HANDLE                 hdl
)
```

DESCRIPTION

This 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 function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010SelftestEnable( hdl );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state first.

SEE ALSO

tdrv010exa.c for a programming example.

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

2.3.7 tdrv010SelftestDisable

NAME

tdrv010SelftestDisable – Disable self test facility

SYNOPSIS

```
TDRV010_STATUS tdrv010SelftestDisable
(
    TDRV010_HANDLE          hdl
)
```

DESCRIPTION

This function disables the self test facility of the SJA1000 CAN controller, which was enabled before with the function tdrv010SelftestEnable().

This function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010SelftestDisable( hdl );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state first.

SEE ALSO

tdrv010exa.c for a programming example.

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

2.3.8 tdrv010ListenOnlyEnable

NAME

tdrv010ListenOnlyEnable – Enable listen-only facility

SYNOPSIS

```
TDRV010_STATUS tdrv010ListenOnlyEnable
(
    TDRV010_HANDLE          hdl
)
```

DESCRIPTION

This 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 function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010ListenOnlyEnable( hdl );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state first.

SEE ALSO

tdrv010exa.c for a programming example.

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

2.3.9 tdrv010ListenOnlyDisable

NAME

tdrv010ListenOnlyDisable – Disable listen-only facility

SYNOPSIS

```
TDRV010_STATUS tdrv010ListenOnlyDisable
(
    TDRV010_HANDLE          hdl
)
```

DESCRIPTION

This function disables the self test facility of the SJA1000 CAN controller, which was enabled before with the function FIO_TDRV010_ENABLE_SELFTEST.

This function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

result = tdrv010ListenOnlyDisable( hdl );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state first.

SEE ALSO

tdrv010exa.c for a programming example.

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

2.3.10 tdrv010Setlimit

NAME

tdrv010SetLimit – Disable listen-only facility

SYNOPSIS

```
TDRV010_STATUS tdrv010SetLimit
(
        TDRV010_HANDLE              hdl,
        unsigned char               ErrorLimit
)
```

DESCRIPTION

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

This function will be accepted only in reset mode (BUSOFF). Use function tdrv010Stop() first.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

ErrorLimit

This parameter specifies the new error warning limit.

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

…
```

```
…

/*
** Set Error Warning Limit to 20
*/
result = tdrv010SetLimit( hdl, 20 );
if (result != TDRV010_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_ACCESS	Permission denied. The controller is currently in BUS ON state. Please enter the BUS OFF state first.

SEE ALSO

tdrv010exa.c for a programming example.

SJA1000 Product Specification Manual – *6.4.10 ERROR WARNING LIMIT REGISTER (EWLR)*

2.3.11 tdrv010CanReset

NAME

tdrv010CanReset – Set CAN controller into reset or operating mode

SYNOPSIS

```
TDRV010_STATUS tdrv010CanReset
(
    TDRV010_HANDLE          hdl,
    unsigned char           CanReset
)
```

DESCRIPTION

This function sets the certain CAN controller in reset or operating mode. After driver startup, the CAN controllers are configured to operating mode.

This function is only available for TPMC310 devices.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

CanReset

This parameter specifies the controller operating mode.

Value	**Description**
TDRV010_CANRESET_RESET	Set the certain CAN channel into reset mode
TDRV010_CANRESET_OPERATING	Set the certain CAN channel into operating mode

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;


…
```

```
…

/*
** Set Controller into operating mode
*/
result = tdrv010CanReset( hdl, TDRV010_CANRESET_OPERATING );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_NOTSUP	Function not supported by the device.

SEE ALSO

TPMC310 User Manual

2.3.12 tdrv010CanSel

NAME

tdrv010CanSel – Set CAN transceiver into silent or operating mode

SYNOPSIS

```
TDRV010_STATUS tdrv010CanSel
(
      TDRV010_HANDLE            hdl,
      unsigned char             CanSel
)
```

DESCRIPTION

This function sets the certain CAN transceivers into silent or operating mode. After driver startup, the CAN transceivers are configured to operating mode.

This function is only available for TPMC310 devices.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

CanSel

This parameter specifies the controller operating mode.

Value	**Description**
TDRV010_CANSEL_SILENT	Set the certain CAN channel into silent mode
TDRV010_CANSEL_OPERATING	Set the certain CAN channel into operating mode

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

…
```

```
…

/*
** Set Transceiver into operating mode
*/
result = tdrv010CanSel( hdl, TDRV010_CANSEL_OPERATING );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_NOTSUP	Function not supported by the device.

SEE ALSO

TPMC310 User Manual

2.3.13 tdrv010CanInt

NAME

tdrv010CanInt – Enable or disable CAN controller interrupts

SYNOPSIS

```
TDRV010_STATUS tdrv010CanInt
(
        TDRV010_HANDLE                  hdl,
        unsigned char                   CanInt
)
```

DESCRIPTION

This function enables or disables certain CAN controller interrupts. After driver startup, the CAN controller interrupts are enabled.

This function is only available for TPMC310 devices.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

CanInt

This parameter specifies the controller operating mode.

Value	Description
TDRV010_CANINT_ENABLE	Enable interrupt of a certain CAN channel
TDRV010_CANINT_DISABLE	Disable interrupt of a certain CAN channel

EXAMPLE

```
#include “tdrv010api.h”

TDRV010_HANDLE     hdl;
TDRV010_STATUS     result;

…
```

```
…

/*
** Enable CAN controller interrupts
*/
result = tdrv010CanInt( hdl, TDRV010_CANINT_ENABLE );
if (result != TDRV010_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV010_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV010_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV010_ERR_NOTSUP	Function not supported by device.

SEE ALSO

TPMC310 User Manual

3 Appendix

3.1 Enable RTP-Support

Using TDRV010 devices tunneled from Real Time Processes (RTPs) is implemented. For this the "TEWS TDRV010 IOCTL command validation" must be enabled in system configuration.

The API source file "tdrv010api.c" must be added to the RTP-Project directory and built together with the RTP-application.

The definition of TVXB_RTP_CONTEXT must be added to the project, which is used to eliminate kernel headers, values and functions from the used driver files.

Find more detailed information in "TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide".

All legacy functions, functions for version compatibility and debugging functions are not usable from RTPs.

3.2 Debugging and Diagnostic

The TDRV010 device driver provides a function and debug statements to display versatile information of the driver installation and status on the debugging console.

If the VxBus driver is used, the TDRV010 show routine is included in the driver by default and can be called from the VxWorks shell. If this function is not needed or program space is rare the function can be removed from the code by un-defining the macro INCLUDE_TDRV010_SHOW in tdrv010drv.c

The tdrv010Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TDRV010 modules and device statistics.

If TDRV010 modules were probed but no devices were created it may helpful to enable debugging code inside the driver code by defining the macro TDRV010_DEBUG in tdrv010drv.c.

In contrast to VxBus TDRV010 devices, legacy TDRV010 devices must be created “manually”. This will be done with the first call to the tdrv010Open API function.

```
-> tdrv010Show
Probed Modules:
    [0] TPMC810: Bus=4, Dev=1, DevId=0x032a, VenId=0x1498, Init=OK, vxDev=0x498218

Associated Devices:
    [0] TPMC810: /tdrv010/0/0 /tdrv010/0/1

Device Statistics:
    /tdrv010/0/0:
        open count = 0
        interrupt count = 1
        bus off count = 0
        receive count = 0
        transmit count = 1
        object overrun = 0
        fifo overrun = 0
        timing value = 0x2f43
        Acceptance Code = 0x0
        Acceptance Mask = 0xffffffff
        SingleFilter = Yes
    /tdrv010/0/1:
        open count = 0
        interrupt count = 1
        bus off count = 0
        receive count = 1
        transmit count = 0
        object overrun = 0
        fifo overrun = 0
        timing value = 0x2f43
        Acceptance Code = 0x0
        Acceptance Mask = 0xffffffff
        SingleFilter = Yes
```