The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV010-SW-65

Windows Device Driver

Isolated 2 x CAN Bus

Version 2.1.x

User Manual

Issue 2.1.0

September 2020

powerBridge Computer

Ehlbeek 15a
30938 Burgwedel
fon 05139-9980-0
fax 05139-9980-49

www.powerbridge.de
info@powerbridge.de

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7 25469 Halstenbek, Germany
9 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19
il: info@tews.com www.tews.com

TDRV010-SW-65

Windows Device Driver

Isolated 2 x CAN Bus

Supported Modules:
TPMC310
TPMC810

Issue	Description	Date
1.0.0	First Issue	January 1, 2005
1.0.1	IOCTL_TDRV010_READ function description added	February 14, 2006
1.0.2	New Address TEWS LLC	February 28, 2007
1.0.3	General Revision, Return value of CloseHandle() corrected	April 7, 2008
1.0.4	Files moved to subdirectory	June 23, 2008
2.0.0	Windows 7, 64-bit support added, API added, Address TEWS LLC removed	January 11, 2011
2.1.0	Description of installation extracted to separate manual, Chapter Device Driver Programming removed. General Revision of document	September 15, 2020

Table of Contents

1 Introduction

The TDRV010-SW-65 Windows device driver is a kernel mode driver which allows the operation of the supported hardware module on an Intel or Intel-compatible Windows operating system.

The TDRV010-SW-65 device driver supports the following features:

- Read received messages from input FIFO
- Send a message
- Set channel Bus On/Off
- Configure Listen-Only mode On/Off
- Configure Selftest mode On/Off
- Extended and Standard Identifiers
- Configure Bitrate
- Configure Receive Mask
- Flush receive FIFO
- Read CAN status
- PLD Functions (external CAN Controller Reset, Silent Mode and Interrupt Control)(TPMC310 only)

The TDRV010-SW-65 device driver supports the modules listed below:

TPMC310	Isolated 2 x CAN Bus (Conduction Cooled) (64 pin connector for Back-IO)	(PMC)
TPMC810	Isolated 2 x CAN Bus (2x SUBD 9 pin connectors for Front-Panel I/O) (64 pin connector for Back-I/O)	(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 TDRV010 devices it is recommended to read the manuals listed below.

TPMC310/TPMC810 User manual
SJA1000 Product Specification
Installation-Guide on Distribution Media

2 Installation

Following files are located in directory TDRV010-SW-65 on the distribution media:

driver*	Directory containing driver files
example\tdrv010exa.c	Example application source
api\tdrv010api.h	Application Programming Interface header
TDRV010-SW-65-2.1.0.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history
tdrv010exa.exe	Example application (executable for quick start)

Copy all required Header and API files into your desired project directory.

For device driver installation execute the application “Setup” on the distribution media. For more information refer to the Installation-Guide, which is also part of the distribution media.

3 Driver Configuration

3.1 Message FIFO Configuration

After Installation of the TDRV010 Device Driver the number of sequential CAN Messages is limited to 100 without the need to read from the certain device.

If the default values are not suitable the configuration can be changed by modifying the registry, for instance with regedit.

To change the number of queue entries the following value must be modified.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tdrv010\Parameters\FIFODepth

Valid values are in range between 1 … 1024

4 API Documentation

4.1 General Functions

4.1.1 tdrv010Open

NAME

tdrv010Open – opens a device

SYNOPSIS

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

DESCRIPTION

Before I/O can be performed to a device, a file descriptor must be opened by a call to this function.

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;

/*
** open file descriptor to device
*/
FileDescriptor = tdrv010Open("\\\\.\\TDRV010_1" );
if (FileDescriptor == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. To get extended error information, call ***GetLastError***.

ERROR CODES

All error codes are standard error codes set by the I/O system.

4.1.2 tdrv010Close

NAME

tdrv010Close – closes a device

SYNOPSIS

```
int tdrv010Close
(
      TDRV010_HANDLE          FileDescriptor
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

FileDescriptor

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

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

/*
** close file descriptor to device
*/
result = tdrv010Close(FileDescriptor);
if (result < 0)
{
     /* handle close error */
}
```

RETURNS

Zero, or a negative error code.

ERROR CODES

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

4.2 Device Access Functions

4.2.1 tdrv010Read

NAME

tdrv010Read – read a CAN message from device

SYNOPSIS

```
int tdrv010Read
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        ULONG               timeout,
        ULONG               *pIdentifier,
        UCHAR               *pIOFlags,
        UCHAR               *pStatus,
        int                 *pLength,
        UCHAR               *pData
)
```

DESCRIPTION

This function reads a CAN message from a specified device. If no message has been received, the function will wait until a message is received, or the function will timeout after the specified time.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

timeout

This argument specifies the maximum time (in seconds) the function is willing to wait for an incoming message. The time is specified in seconds.

pIdentifier

This parameter points to buffer where the message identifier of the received message will be stored to.

pIOFlags

This parameter points to buffer where the I/O flag of the received message will be stored to. The following flags may be set:

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 points to buffer where the status information about overrun condition, either in the CAN controller or intermediate software FIFO, will be stored to.

Value	Description
TDRV010_SUCCESS	No messages lost
TDRV010_FIFO_OVERRUN	One or more messages have been 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 have been 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 points to buffer where the data length of the received message data in bytes will be stored to. The returned length will always be 0…8.

pData

This parameter points to a buffer where the received data bytes will stored to. This buffer must have a length of at least 8 byte. Data[0] receives the first data byte, Data[1] receives the second data byte and so on. The number of valid bytes is specified by *pLength*.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result, i;
ULONG              identifier;
UCHAR              IOFlags;
UCHAR              msgStatus;
int                dataLen;
UCHAR              dataBuf[8];

…
```

```
…

/*
** Read a CAN message from channel 2
**   - timeout after 10 seconds
*/
result = tdrv010Read (  FileDescriptor,
                        2,                  /* channel */
                        10,                 /* timeout */
                        &identifier,
                        &IOFlags,
                        &msgStatus,
                        &dataLen,
                        dataBuf);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
     printf("%s %s Identifier = %ld\n",
            (IOFlags & TDRV010_EXTENDED) ? "Extd" : "Stnd",
            (IOFlags & TDRV010_REMOTE_FRAME) ? "Remote Msg - " : "Msg - ",
            identifier);
     printf("%d data bytes received\n", dataLen);
     for( i = 0; i < dataLen; i++ )
     {
          printf("%02X ", dataBuf[i]);
     }
     printf("\n")
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the read/write buffer is too small, or a buffer pointer is NULL.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_FILE_OFFLINE	The channel must be in BUS ON state to receive messages.
ERROR_NETWORK_BUSY	There is already another job waiting for message reception on this channel.
ERROR_OPERATION_ABORTED	The job has been canceled by Windows.
ERROR_SEM_TIMEOUT	The specified timeout time has expired without receiving a message.

Other returned error codes are system error conditions.

4.2.2 tdrv010ReadNoWait

NAME

tdrv010ReadNoWait – read a CAN message from device (return immediately)

SYNOPSIS

```
int tdrv010ReadNoWait
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        ULONG               *pIdentifier,
        UCHAR               *pIOFlags,
        UCHAR               *pStatus,
        int                 *pLength,
        UCHAR               *pData
)
```

DESCRIPTION

This function reads a CAN message from a specified device. This function will return immediately, either if a message is available or not. If no message has been received the function will return an error.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

pIdentifier

This parameter points to buffer where the message identifier of the received message will be stored to.

pIOFlags

This parameter points to buffer where the I/O flag of the received message will be stored to. The following flags may be set:

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 points to buffer where the status information about overrun condition, either in the CAN controller or intermediate software FIFO, will be stored to.

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 points to buffer where the data length of the received message data in bytes will be stored to. The returned length will always be 0…8.

pData

This parameter points to a buffer where the received data bytes will stored to. This buffer must have a length of at least 8 byte. Data[0] receives the first data byte, Data[1] receives the second data byte and so on. The number of valid bytes is specified by *pLength*.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result, i;
ULONG              identifier;
UCHAR              IOFlags;
UCHAR              msgStatus;
int                dataLen;
UCHAR              dataBuf[8];

...
```

```
...

/*
** Read a CAN message from channel 2
*/
result = tdrv010ReadNoWait( FileDescriptor,
                            2,                  /* channel */
                            &identifier,
                            &IOFlags,
                            &msgStatus,
                            &dataLen,
                            dataBuf);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
     printf("%s %s Identifier = %ld\n",
            (IOFlags & TDRV010_EXTENDED) ? "Extd" : "Stnd",
            (IOFlags & TDRV010_REMOTE_FRAME) ? "Remote Msg - " : "Msg - ",
            identifier);
     printf("%d data bytes received\n", dataLen);
     for( i = 0; i < dataLen; i++ )
     {
          printf("%02X ", dataBuf[i]);
     }
     printf("\n")
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the read/write buffer is too small, or a buffer pointer is NULL.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_FILE_OFFLINE	The channel must be in BUS ON state to receive messages.
ERROR_MR_MID_NOT_FOUND	There is no message in the FIFO buffer.
ERROR_NETWORK_BUSY	There is already another job waiting for message reception on this channel.
ERROR_OPERATION_ABORTED	The job has been canceled by Windows.

Other returned error codes are system error conditions.

4.2.3 tdrv010Write

NAME

tdrv010Write – send a CAN message to device

SYNOPSIS

```
int tdrv010Write
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        ULONG               timeout,
        ULONG               identifier,
        UCHAR               IOFlags,
        int                 length,
        UCHAR               *pData
)
```

DESCRIPTION

This function sends a CAN message to a specified device and waits until it is send. If the message cannot be sent within a specified timeout time, the function will return and indicate an error.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

timeout

This argument specifies the maximum time (in seconds) the function is willing to wait for a successful sending (acknowledge) for the sent message.

identifier

This argument specifies the message identifier of the message.

IOFlags

This parameter specifies the I/O flags for the message. The following flags may be set:

Value	Description
TDRV010_EXTENDED	Set if the transmit message is an extended message frame. Reset for standard message frames.
TDRV010_REMOTE_FRAME	Set if the transmit message is a remote transmission request (RTR) frame.
TDRV010_SINGLE_SHOT	There will be no retry sending the message
TDRV010_SELF_RECEPTION	The message will be able to be received on the sending device.

length

This parameter specifies the data length of the message data in bytes. The length of the message can be 0…8 byte.

pData

This parameter points to a buffer where the message data must be stored. The number of used bytes must be specified in *length*.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;
UCHAR              dataBuf[8] = "123456";

/*
** Send a CAN message on channel 2
**   - extended identifier (42)
**   - timout after 5 seconds
*/
result = tdrv010Write ( FileDescriptor,
                        2,            /* channel */
                        5,            /* timeout */
                        42,
                        TDRV010_EXTENDED,
                        6,
                        dataBuf);

…
```

```
...

if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the read/write buffer is too small, or a buffer pointer is NULL.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_FILE_OFFLINE	The channel must be in BUS ON state to receive messages.
ERROR_NETWORK_BUSY	There is already another job waiting for message reception on this channel.
ERROR_OPERATION_ABORTED	The job has been canceled by Windows.
ERROR_SEM_TIMEOUT	The specified timeout time has expired without receiving a message.

Other returned error codes are system error conditions.

4.2.4 tdrv010WriteNoWait

NAME

tdrv010WriteNoWait – send a CAN message to device (return immediately)

SYNOPSIS

```
int tdrv010WriteNoWait
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        ULONG               identifier,
        UCHAR               IOFlags,
        int                 length,
        UCHAR               *pData
)
```

DESCRIPTION

This function starts sending a CAN message to a specified device and returns. The function will return immediately and will not wait until the message is send.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

identifier

This argument specifies the message identifier of the message.

IOFlags

This parameter specifies the I/O flags for the message. The following flags may be set:

Value	Description
TDRV010_EXTENDED	Set if the transmit message is an extended message frame. Reset for standard message frames.
TDRV010_REMOTE_FRAME	Set if the transmit message is a remote transmission request (RTR) frame.
TDRV010_SINGLE_SHOT	There will be no retry sending the message
TDRV010_SELF_RECEPTION	The message will be able to be received on the sending device.

length

This parameter specifies the data length of the message data in bytes. The length of the message can be 0…8 byte.

pData

This parameter points to a buffer where the message data must be stored to. The number of used bytes must be specified in *length*.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;
UCHAR              dataBuf[8] = "123456";

/*
** Send a CAN message on channel 1
**   - standard identifier (42)
*/
result = tdrv010WriteNoWait( FileDescriptor,
                             1,            /* channel */
                             42,
                             TDRV010_STANDARD,
                             6,
                             dataBuf);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the read/write buffer is too small, or a buffer pointer is NULL.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_FILE_OFFLINE	The channel must be in BUS ON state to receive messages.
ERROR_NETWORK_BUSY	There is already another job waiting for message reception on this channel.
ERROR_OPERATION_ABORTED	The job has been canceled by Windows.

Other returned error codes are system error conditions.

4.2.5 tdrv010SetBitTiming

NAME

tdrv010SetBitTiming – configure the bit-timing of the specified device

SYNOPSIS

```
int tdrv010SetBitTiming
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        USHORT              timingValue,
        BOOLEAN             useThreeSamples
)
```

DESCRIPTION

This function configures a new bit-timing for the specified CAN channel.

This function must be executed in BUS OFF state.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to configure. Allowed values are 1 for channel 1 and 2 for channel 2.

timingValue

This argument specifies 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 60 Kbit per second and 1 Mbit per second. The include file 'tdrv010api.h' contains predefined transfer rate symbols (TDRV010_100KBIT … TDRV010_1MBIT).
For other transfer rates please follow the instructions of the *SJA1000 Product Specification*, which is also part of the corresponding hardware engineering documentation.

useThreeSamples

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

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

/*
** Configure channel 2 for 250Kbit/s
*/
result = tdrv010SetBitTiming (   FileDescriptor,
                                 2,                  /* channel */
                                 TDRV010_250KBIT,
                                 FALSE);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.6 tdrv010SetFilter

NAME

tdrv010SetFilter – configure the acceptance filter

SYNOPSIS

```
int tdrv010SetFilter
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        BOOLEAN             singleFilter,
        ULONG               acceptanceCode,
        ULONG               acceptanceMask
)
```

DESCRIPTION

This function configures the acceptance filtering of the specified CAN channel.

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 must be executed in BUS OFF state.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to configure. Allowed values are 1 for channel 1 and 2 for channel 2.

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     FileDescriptor;
int                result;

/*
** Configure channel 2 to accept all incoming messages
*/
result = tdrv010SetFilter (  FileDescriptor,
                             2,                  /* channel */
                             FALSE,
                             0x00000000,
                             0xFFFFFFFF);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.7 tdrv010Start

NAME

tdrv010Start – sets the CAN channel to bus on mode

SYNOPSIS

```
int tdrv010Start
(
      TDRV010_HANDLE     FileDescriptor,
      UCHAR              canChan
)
```

DESCRIPTION

This function starts the specified CAN channel and set it to bus on mode.

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 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.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to start. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

...
```

```
...

/*
** Start up channel 2
*/
result = tdrv010Start ( FileDescriptor,
                        2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_IO_DEVICE	Error during Bus On sequence.

Other returned error codes are system error conditions.

4.2.8 tdrv010Stop

NAME

tdrv010Stop – sets the CAN channel to bus off mode

SYNOPSIS

```
int tdrv010Stop
(
    TDRV010_HANDLE    FileDescriptor,
    UCHAR             canChan
)
```

DESCRIPTION

This function stops the specified CAN channel and set it to bus off mode.

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

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to stop. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

...
```

```
...

/*
** Stop channel 2
*/
result = tdrv010Stop (  FileDescriptor,
                        2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_IO_DEVICE	Error during Bus Off sequence.

Other returned error codes are system error conditions.

4.2.9 tdrv010FlushReceiveFifo

NAME

tdrv010FlushReceiveFifo – flush receive buffer of CAN channel

SYNOPSIS

```
int tdrv010FlushReceiveFifo
(
      TDRV010_HANDLE     FileDescriptor,
      UCHAR              canChan
)
```

DESCRIPTION

This function flushes the receive buffer of the specified channel, and removes all previously received messages from the FIFO.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to stop. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

...
```

```
...

/*
** Flush receive FIFO of channel 2
*/
result = tdrv010FlushReceiveFifo(FileDescriptor,
                                 2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.

Other returned error codes are system error conditions.

4.2.10 tdrv010GetControllerStatus

NAME

tdrv010GetControllerStatus – returns the CAN controller state

SYNOPSIS

```
int tdrv010GetControllerStatus
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        TDRV010_STATUS      *pCANStatus
)
```

DESCRIPTION

This function returns the actual contents of several CAN controller registers for diagnostic purposes in an application supplied buffer (*TDRV010_STATUS*).

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

pCANStatus

This argument points to a buffer (*TDRV010_STATUS*) where the current status will be filled in.

```
typedef struct
{
        UCHAR       channel;
        UCHAR       ArbitrationLostCapture;
        UCHAR       ErrorCodeCapture;
        UCHAR       TxErrorCounter;
        UCHAR       RxErrorCounter;
        UCHAR       ErrorWarningLimit;
        UCHAR       StatusRegister;
        UCHAR       ModeRegister;
        UCHAR       RxMessageCounterMax;
} TDRV010_STATUS, *PTDRV010_STATUS;
```

channel

This parameter is not used by this function. Set to same value as *canChan*.

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.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;
TDRV010_STATUS     statusBuf;

...
```

```
...

/*
** Read a CAN controller status of channel 2
*/
result = tdrv010GetControllerStatus ( FileDescriptor,
                                       channel,
                                       &statusBuf);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
     printf ( "\nArbitration lost capture register  = 0x%02X\n",
              statusBuf.ArbitrationLostCapture);
     printf ( "Error code capture register          = 0x%02X\n",
              statusBuf.ErrorCodeCapture);
     printf ( "TX error counter register            = 0x%02X\n",
              statusBuf.TxErrorCounter);
     printf ( "RX error counter register            = 0x%02X\n",
              statusBuf.RxErrorCounter);
     printf ( "Error warning limit register         = 0x%02X\n",
              statusBuf.ErrorWarningLimit);
     printf ( "Status register                      = 0x%02X\n",
              statusBuf.StatusRegister);
     printf ( "Mode register                        = 0x%02X\n",
              statusBuf.ModeRegister);
     printf ( "Peak value RX message counter        = %d\n",
              statusBuf.RxMessageCounterMax);
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the read/write buffer is too small, or a buffer pointer is NULL.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.

Other returned error codes are system error conditions.

4.2.11 tdrv010SelftestEnable

NAME

tdrv010SelftestEnable – enable the self-test facility of CAN channel

SYNOPSIS

```
int tdrv010SelftestEnable
(
      TDRV010_HANDLE     FileDescriptor,
      UCHAR              canChan
)
```

DESCRIPTION

This function enables the self-test facility of the SJA1000 CAN controller for the specified channel.

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.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

…
```

```
...

/*
** Enable self-test for channel 2
*/
result = tdrv010SelftestEnable ( FileDescriptor,
                                  2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.12 tdrv010SelftestDisable

NAME

tdrv010SelftestDisable – disable the self-test facility of CAN channel

SYNOPSIS

```
int tdrv010SelftestDisable
(
      TDRV010_HANDLE     FileDescriptor,
      UCHAR              canChan
)
```

DESCRIPTION

This function disables the self-test facility of the SJA1000 CAN controller for the specified channel.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

...
```

```
...

/*
** Disable self-test for channel 2
*/
result = tdrv010SelftestDisable (FileDescriptor,
                                 2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.13 tdrv010ListenOnlyEnable

NAME

tdrv010ListenOnlyEnable – enable the listen only facility of CAN channel

SYNOPSIS

```
int tdrv010ListenOnlyEnable
(
      TDRV010_HANDLE     FileDescriptor,
      UCHAR              canChan
);
```

DESCRIPTION

This function enables the listen-only facility of the SJA1000 CAN controller for the specified channel.

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'.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

...
```

```
...

/*
** Enable listen-only for channel 2
*/
result = tdrv010ListenOnlyEnable(FileDescriptor,
                                 2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.14 tdrv010ListenOnlyDisable

NAME

tdrv010ListenOnlyDisable – disable the listen-only facility of CAN channel

SYNOPSIS

```
int tdrv010ListenOnlyDisable
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan
)
```

DESCRIPTION

This function disables the listen-only facility of the SJA1000 CAN controller for the specified channel.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to use. Allowed values are 1 for channel 1 and 2 for channel 2.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE      FileDescriptor;
int                 result;

...
```

```
...

/*
** Disable listen-only for channel 2
*/
result = tdrv010ListenOnlyDisable (   FileDescriptor,
                                      2);           /* channel */
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.15 tdrv010SetLimit

NAME

tdrv010SetLimit – configure error warning limit of the specified device

SYNOPSIS

```
int tdrv010SetLimit
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        UCHAR               errorLimit
)
```

DESCRIPTION

This function sets a new error warning limit in the corresponding CAN controller register of the specified CAN channel.

This function must be executed in BUS OFF state.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to configure. Allowed values are 1 for channel 1 and 2 for channel 2.

errorLimit

This parameter specifies the new warning limit. Allowed values are between 0 and 255.

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE     FileDescriptor;
int                result;

/*
** Set error warning limit to 100 for channel 2
*/
result = tdrv010SetLimit (   FileDescriptor,
                             2,            /* channel */
                             100);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_OPEN_FILES	The channel must be in BUS OFF state to execute this function.

Other returned error codes are system error conditions.

4.2.16 tdrv010CanReset

NAME

tdrv010CanReset – set CAN channel to reset / operating mode

SYNOPSIS

```
int tdrv010CanReset
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        BOOLEAN             canReset
)
```

DESCRIPTION

This function sets the specified CAN controller in reset or operating mode by controlling the external controller reset pin of the specified CAN channel.

This function is only available for TPMC310.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to configure. Allowed values are 1 for channel 1 and 2 for channel 2.

canReset

This parameter specifies if controller shall be set in reset (TRUE) or in operating mode (FALSE).

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE      FileDescriptor;
int                 result;

…
```

```
...

/*
** Set channel 1 to operating mode and channel 2 into reset
*/
result = tdrv010CanReset (   FileDescriptor,
                             1,             /* channel */
                             FALSE);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}

result = tdrv010CanReset (   FileDescriptor,
                             2,             /* channel */
                             TRUE);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_IO_DEVICE	Unable to reinitialize the certain CAN controller.
ERROR_INVALID_FUNCTION	This function is only supported by TPMC310 modules

Other returned error codes are system error conditions.

4.2.17 tdrv010CanSel

NAME

tdrv010CanSel – set CAN channel to silent mode

SYNOPSIS

```
int tdrv010CanSel
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        BOOLEAN             canSel
)
```

DESCRIPTION

This function sets the specified CAN channel in silent or operating mode.

This function is only available for TPMC310.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to configure. Allowed values are 1 for channel 1 and 2 for channel 2.

canSel

This parameter specifies if controller shall be in silent mode (TRUE) or in operating mode (FALSE).

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE      FileDescriptor;
int                 result;

...
```

```
...

/*
** Set channel 1 to silent mode
*/
result = tdrv010CanSel( FileDescriptor,
                        1,            /* channel */
                        TRUE);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_INVALID_FUNCTION	This function is only supported by TPMC310 modules

Other returned error codes are system error conditions.

4.2.18 tdrv010CanInt

NAME

tdrv010CanInt – enable or disable the global interrupt of the CAN controller

SYNOPSIS

```
int tdrv010CanInt
(
        TDRV010_HANDLE      FileDescriptor,
        UCHAR               canChan,
        BOOLEAN             canInt
)
```

DESCRIPTION

This function enables or disables the global interrupt of the specified CAN controller.

This function is only available for TPMC310.

PARAMETERS

FileDescriptor

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

canChan

This argument specifies the channel to configure. Allowed values are 1 for channel 1 and 2 for channel 2.

canInt

This parameter specifies if the global interrupt of the CAN controller shall be enabled (TRUE) or disabled (FALSE).

EXAMPLE

```
#include "tdrv010api.h"

TDRV010_HANDLE      FileDescriptor;
int                 result;

...
```

```
...

/*
** Enable global interrupt on channel 2
*/
result = tdrv010CanInt( FileDescriptor,
                        2,            /* channel */
                        TRUE);
if (result != 0)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURN VALUE

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

ERROR CODES

Error code	Description
ERROR_INVALID_PARAMETER	This error will be returned if the size of the user buffer is too small.
ERROR_FILE_NOT_FOUND	Illegal channel number specified.
ERROR_INVALID_FUNCTION	This function is only supported by TPMC310 modules

Other returned error codes are system error conditions.