The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV011-SW-65

Windows Device Driver

Extended CAN

Version 3.0.x

User Manual

Issue 3.0.1

July 2013

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

TDRV011-SW-65

Windows Device Driver

Extended CAN

Supported Modules:
TPMC316
TPMC816
TPMC901

Issue	Description	Date
1.0.0	First Issue	June 06, 2007
1.0.1	Files moved to subdirectory	June 23, 2008
1.0.2	General revision	October 14, 2009
1.0.3	Added programming hints for IOCTL_TDRV011_SETFILTER and IOCTL_TDRV011_DEF_RX_BUF	February 11, 2010
2.0.0	General Revision, Windows 7 and 64-bit Support, Implementation of API	June 16, 2011
3.0.0	New Parameter List for Function tdrv011UpdateReceiveMsgObj	August 16, 2013
3.0.1	General revision	July 8, 2014

Table of Contents

1 Introduction

The TDRV011-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 TDRV011-SW-65 device driver supports the following features:

- Transmission and receive of standard and extended Identifiers
- Up to 14 receive message queues with user defined size
- Variable allocation of receive message objects to receive queues
- Separate job queues for each receive queue and transmission buffer message object
- Standard bit rates from 20 kbit up to 1.0 Mbit and user defined bit rates
- Message acceptance filtering
- Definition of receive and remote buffer message objects

The TDRV011-SW-65 device driver supports the modules listed below:

TPMC316	2 Channel extended CAN (isolated)	(PMC, Conduction Cooled)
TPMC816	2/1 Channel extended CAN (isolated)	(PMC)
TPMC901	6/4/2 Channel extended CAN	(PMC)

In this document all supported modules and devices will be called TDRV011. Specials for certain devices will be advised.

To get more information about the features and use of TDRV011 devices it is recommended to read the manuals listed below.

TPMC316, TPMC816 or TPMC901 User manual
Intel 82527 Architectural Overview

2 Installation

Following files are located in directory TDRV011-SW-65 on the distribution media:

i386\	Directory containing driver files for 32bit Windows versions
amd64\	Directory containing driver files for 64bit Windows versions
installer_32bit.exe	Installation tool for 32bit systems (Windows XP or later)
installer_64bit.exe	Installation tool for 64bit systems (Windows XP or later)
tdrv011.inf	Windows installation script
tdrv011.h	Header file with IOCTL codes and structure definitions
api\tdrv011api.h	API include file
api\tdrv011api.c	API source file
example\tdrv011exa.c	Example application
TDRV011-SW-65-3.0.1.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation (Windows 7 / XP)

This chapter describes how to install the TDRV011-SW-65 Device Driver on a Windows 7 (32bit or 64bit) or Windows XP (32bit) operating system.

Depending on the operating system type, execute the installer binaries for either 32bit or 64bit systems. This will install all required driver files using an installation wizard.

Copy needed files (tdrv011.h, API files) to desired target directory.

After successful installation a device is created for each module found (TDRV011_1, TDRV011_2 ...).

2.2 Confirming Driver Installation

To confirm that the driver has been properly loaded, perform the following steps:

1. Open the Windows Device Manager:

   a. For Windows XP, open the "***Control Panel***" from "***My Computer***" and click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.

   b. For Windows 7, open the "***Control Panel***" from "***My Computer***" and then click the "***Device Manager***" entry.

2. Click the "**+**" in front of "***Embedded I/O***".
   The driver "***TEWS TECHNOLOGIES - TDRV011 (Multi Channel Extended CAN) (…)***" should appear for each installed device.

3 Driver Configuration

The driver allows individual adaptations of buffers by changing parameters in the windows registry. All registry keys described below can be found and modified in the registry path:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tdrv011\Parameters

After modification of any of the registry keys the driver (or the system) needs to be restarted to make the changes applicable.

3.1 Number of Receive Queues

The number of receive queues can be modified. This allows an adaptation for applications which should use presorting of messages by message objects.

To change the number of receive queues the value of *NumRxQueues* in registry path must be modified.

Default value:	2
Valid value range:	1…14

3.2 Depth of Receive Queue

The depth of the receive queue may be adapted if the application may not read data or is blocked by another application for a while, but there are still incoming messages that must be handled. Increasing the value will allow storing more messages in the receive queue(s).

To change the depth of receive queues the value of *FIFODepth* in registry path must be modified.

Default value:	100
Valid value range:	1…1000

3.3 Transmit Message Object

This parameter specifies the message object used for transmit. The selected object cannot be defined for receive or remote and it cannot be released.

To change the transmit message object the value of *TransmitObject* in registry path must be modified.

Default value:	1
Valid value range:	1…14

4 API Documentation

4.1 General Functions

4.1.1 tdrv011Open

NAME

tdrv011Open – opens a device.

SYNOPSIS

```
TDRV011_HANDLE tdrv011Open
(
        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 "tdrv011api.h"

TDRV011_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tdrv011Open("\\\\.\\TDRV011_1");
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

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

4.1.2 tdrv011Close

NAME

tdrv011Close – closes a device.

SYNOPSIS

```
int tdrv011Close
(
      TDRV011_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 "tdrv011api.h"

TDRV011_HANDLE     hdl;
TDRV011_STATUS     result;

/*
** close file descriptor to device
*/
result = tdrv011Close( hdl );
if (result != TDRV011_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

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

4.2 Device Access Functions

4.2.1 tdrv011Write

NAME

tdrv011Write – write a CAN message

SYNOPSIS

```
TDRV011_STATUS tdrv011Write
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 timeout,
    unsigned int        identifier,
    unsigned int        extMsgFlag,
    int                 length,
    unsigned char       *pData
)
```

DESCRIPTION

This function writes a messages to the specified device for subsequent transmission on the CAN bus. The request will be blocked until the message was send or an error occurs.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message shall be send. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

timeout

This argument specifies the time (in milliseconds, one second granularity) the function is willing to wait for a completion of message transfer.

identifier

This argument specifies the message identifier of the message.

extMsgFlag

This argument specifies if the message shall be send in standard or extended CAN message format. One of the following flags must be set:

Value	Description
TDRV011_STANDARD_IDENTIFIER	Set if the message shall be send as a standard CAN message frame.
TDRV011_EXTENDED_IDENTIFIER	Set if the message shall be send as an extended CAN message frame.

length

This argument specifies the data length of the message data stored in the data buffer (*pData*). A valid length is 0…8.

pData

This argument points to a buffer where the write data bytes are stored to. Data[0] contains the first data byte to send, Data[1] contains the second data byte and so on. The number of valid bytes must be specified by l*ength*.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Send a CAN message on channel 2
*/
result = tdrv011Write ( hdl,
                        2,              /* channel */
                        5000,           /* 5 seconds */
                        1234,           /* message identifier: 1234 */
                        TDRV011_EXTENDED_IDENTIFIER,
                        5,              /* number of valid data bytes */
                        "Hello");
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.
TDRV011_ERR_NETDOWN	The channel is stopped (BUS OFF), but it must be in BUS ON state to receive messages.
TDRV011_ERR_NOMEM	An error occurred when allocating memory for the request.
TPMC011_ERR_TIMEOUT	The specified timeout time has expired before the message was send.

Other returned error codes are system error conditions.

4.2.2 tdrv011Read

NAME

tdrv011Read – read a CAN message from device

SYNOPSIS

```
TDRV011_STATUS tdrv011Read
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 rcvQueue,
    int                 timeout,
    unsigned int        flags,
    unsigned int        *pIdentifier,
    unsigned int        *pExtMsgFlag,
    int                 *pLength,
    unsigned char       *pData,
    unsigned int        *pOverrunState
)
```

DESCRIPTION

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

Before the driver can receive CAN messages it's necessary to define at least one receive message object.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which a message shall be received. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

rcvQueue

This argument specifies the receive queue number to use. Allowed values are 1 up to the last configured receive queue number.

timeout

This argument specifies the time (in milliseconds, one second granularity) the function is willing to wait for an incoming message.

flags

This argument specifies special settings for this read function. The specified flags are defined:

Flag	Description
TDRV011_FLUSH_BEFORE_READ	If this flag is set, the specified receive queue will be flushed, before the read request is started. The function will not return messages that have been received before the function has been called.

pIdentifier

This argument is a pointer to an unsigned int variable where the message identifier of the received message will be stored to.

pExtMsgFlag

This argument is a pointer to an unsigned int variable where the function sets a flag if the received message contains a standard or an extended identifier. The following flags may be set:

Value	Description
TDRV011_STANDARD_IDENTIFIER	Set if the received message is a standard CAN message frame.
TDRV011_EXTENDED_IDENTIFIER	Set if the received message is an extended CAN message frame.

pLength

This argument is a pointer to an int variable 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 argument 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*.

pOverrunState

This parameter is a pointer to an unsigned int variable which receives the status information about overrun conditions either in the CAN controller or intermediate software FIFO's.

Value	Description
TDRV011_SUCCESS	No messages lost.
TDRV011_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.
TDRV011_MSGOBJ_OVERRUN	One or more messages have been overwritten in the CAN controller message object because the interrupt latency is too large. Keep in mind Windows isn't a real-time operating system. Use message object 15 (buffered) to receive this time critical CAN messages, reduce the CAN bit rate or upgrade the system speed.
TDRV011_RAW_FIFO_OVERRUN	One or more messages have been overwritten in the FIFO between the interrupt service routine and post-processing in the driver, on lower system priority levels

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result, i;
unsigned int       identifier;
unsigned int       extMsgFlags;
int                dataLen;
unsigned char      dataBuf[8];
unsigned int       overrunState;

…
```

```
…

/*
** Read a CAN message from channel 2, queue 2
**   - timeout after 5 seconds
**   - flush Rx FIFO before read
*/
result = tdrv011Read (  hdl,
                        2,              /* channel */
                        2,              /* receive queue */
                        5000,           /* timeout */
                        TDRV011_FLUSH_BEFORE_READ,
                        &identifier,
                        &extMsgFlags,
                        &dataLen,
                        dataBuf,
                        &overrunState);
if (result != TDRV011_OK)
{
     /* handle error */
}
else
{
     /* successful */
     if(overrunState != TDRV011_SUCCESS)
     {
          printf("<<< message(s) lost >>> \n");
     }
     printf("%s %s Identifier = %ld\n",
            (extMsgFlags & TDRV011_EXTENDED_IDENTIFIER) ? "Extd" : "Stnd",
            identifier);
     printf("%d data bytes received\n", dataLen);
     for( i = 0; i < dataLen; i++ )
     {
          printf("%02X ", dataBuf[i]);
     }
     printf("\n")
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.
TDRV011_ERR_NETDOWN	The channel is stopped (BUS OFF), but it must be in BUS ON state to receive messages.
TDRV011_ERR_BUSY	There is already another job waiting for message reception on the specified queue.
TPMC011_ERR_TIMEOUT	The specified timeout time has expired without receiving a message.

Other returned error codes are system error conditions.

4.2.3 tdrv011ReadNoWait

NAME

tdrv011ReadNoWait – read a CAN message from device (non-blocked)

SYNOPSIS

```
TDRV011_STATUS tdrv011ReadNoWait
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 rcvQueue,
    unsigned int        *pIdentifier,
    unsigned int        *pExtMsgFlag,
    int                 *pLength,
    unsigned char       *pData,
    unsigned int        *pOverrunState
)
```

DESCRIPTION

This function reads a message from a specified device. If no message has been received, the function will return immediately with an appropriate error code.

Before the driver can receive CAN messages it's necessary to define at least one receive message object.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object to be defined. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

rcvQueue

This argument specifies the receive queue number to use. Allowed values are 1 up to the last configured receive queue number.

pIdentifier

This argument is a pointer to an unsigned int variable where the message identifier of the received message will be stored to.

pExtMsgFlag

This argument is a pointer to an unsigned int variable where the function sets a flag if the received message contains a standard or an extended identifier. The following flags may be set:

Value	Description
TDRV011_STANDARD_IDENTIFIER	Set if the received message is a standard CAN message frame.
TDRV011_EXTENDED_IDENTIFIER	Set if the received message is an extended CAN message frame.

pLength

This argument is a pointer to an int variable 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 argument 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*.

pOverrunState

This argument is a pointer to an unsigned int variable which receives the status information about overrun conditions either in the CAN controller or intermediate software FIFO's.

Value	Description
TDRV011_SUCCESS	No messages lost.
TDRV011_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.
TDRV011_MSGOBJ_OVERRUN	One or more messages have been overwritten in the CAN controller message object because the interrupt latency is too large. Keep in mind Windows isn't a real-time operating system. Use message object 15 (buffered) to receive this time critical CAN messages, reduce the CAN bit rate or upgrade the system speed.
TDRV011_RAW_FIFO_OVERRUN	One or more messages have been overwritten in the FIFO between the interrupt service routine and post-processing in the driver, on lower system priority levels

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result, i;
unsigned int       identifier;
unsigned int       extMsgFlags;
int                dataLen;
unsigned char      dataBuf[8];
unsigned int       overrunState;

/*
** Read a CAN message from channel 2, queue 2
**   - timeout after 5 seconds
**   - flush Rx FIFO before read
*/
result = tdrv011ReadNoWait (hdl,
                            2,            /* channel */
                            2,            /* receive queue */
                            &identifier,
                            &extMsgFlags,
                            &dataLen,
                            dataBuf,
                            &overrunState);
if (result != TDRV011_OK)
{
     /* handle error */
}
else
{    /* successful */
     if(overrunState != TDRV011_SUCCESS)
          printf("<<< message(s) lost >>> \n");
     printf("%s %s Identifier = %ld\n",
            (extMsgFlags & TDRV011_EXTENDED_IDENTIFIER) ? "Extd" : "Stnd",
            identifier);
     printf("%d data bytes received\n", dataLen);
     for( i = 0; i < dataLen; i++ )
     {
          printf("%02X ", dataBuf[i]);
     }
     printf("\n")
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.
TDRV011_ERR_NETDOWN	The channel is stopped (BUS OFF), but it must be in BUS ON state to receive messages.
TDRV011_ERR_BUSY	There is already another job waiting for message reception on the specified queue.
TDRV011_ERR_NODATA	No data available.

Other returned error codes are system error conditions.

4.2.4 tdrv011SetFilter

NAME

tdrv011SetFilter – write acceptance filter masks

SYNOPSIS

```
TDRV011_STATUS tdrv011SetFilter
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    unsigned short      globalMaskStandard,
    unsigned int        globalMaskExtended,
    unsigned int        message15Mask
)
```

DESCRIPTION

This function modifies the acceptance filter masks of the specified CAN controller.

The acceptance masks allow message objects to receive messages with a range of message identifiers instead of just a single message identifier. A ‘0’ value means "don't care" or accept a ‘0’ or "1" for that bit position. A value of ‘1’ means that the incoming bit value "must-match" identically to the corresponding bit in the message identifier.

A detailed description of the acceptance filter can be found in the Intel 82527 Architectural Overview - Acceptance Filtering Implications.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object to be defined. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

globalMaskStandard

This argument contains the value for the Global Mask-Standard Register. The Global Mask-Standard Register applies only to messages using the standard CAN identifier. The 11 bit identifier appears in bit position 5..15.

globalMaskExtended

This argument contains the value for the Global Mask-Extended Register. The Global Mask-Extended Register applies only to messages using the extended CAN identifier. The 29 bit identifier appears in bit position 3..31.

message15Mask

This argument contains the value for the Message 15 Mask Register. The Message 15 Mask Register is a local mask for message object 15. The 29 bit identifier appears in bit position 3..31. The Message 15 Mask is "ANDed" with the Global Mask. This means that any bit defined as "don't care" in the Global Mask will automatically be a "don't care" bit for message 15.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Set acceptance filter.
** Message Object 15 shall accept all messages
*/
result = tdrv011SetFilter ( hdl,
                            2,            /* channel */
                            0xFFFF,       /* globalMaskStandard */
                            0xFFFFFFFF,   /* globalMaskExtended */
                            0x00000000);  /* message15Mask */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.5 tdrv011GetFilter

NAME

tdrv011GetFilter – get acceptance filter masks

SYNOPSIS

```
TDRV011_STATUS tdrv011GetFilter
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    unsigned short      *pGlobalMaskStandard,
    unsigned int        *pGlobalMaskExtended,
    unsigned int        *pMessage15Mask
)
```

DESCRIPTION

This function reads the acceptance filter masks from the specified CAN controller.
The acceptance masks allow message objects to receive messages with a range of message identifiers instead of just a single message identifier. A ‘0’ value means "don't care" or accept a ‘0’ or "1" for that bit position. A value of ‘1’ means that the incoming bit value "must-match" identically to the corresponding bit in the message identifier.

A detailed description of the acceptance filter can be found in the Intel 82527 Architectural Overview - Acceptance Filtering Implications.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object to be defined. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

pGlobalMaskStandard

This argument is a pointer to an unsigned short variable where the content of the Global Mask-Standard Register is stored. The 11 bit identifier appears in bit position 5..15.

pGlobalMaskExtended

This argument is a pointer to an unsigned int variable where the content of the Global Mask-Extended Register is stored. The 29 bit identifier appears in bit position 3..31.

pMessage15Mask

This argument is a pointer to an unsigned int variable where the content of the Message 15 Mask Register is stored. The 29 bit identifier appears in bit position 3..31.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
unsigned short     globalMaskStandard;
unsigned int       globalMaskExtended;
unsigned int       message15Mask;
int                result;

/*
** Read acceptance filter.
*/
result = tdrv011GetFilter ( hdl,
                            2,            /* channel */
                            &globalMaskStandard,
                            &globalMaskExtended,
                            &message15Mask);
if (result != TDRV011_OK)
{
     /* handle error */
}
else
{
     printf("Standard Mask: 0x%X\n", globalMaskStandard)
     printf("Extended Mask: 0x%X\n", globalMaskExtended)
     printf("Mask 15:       0x%X\n", message15Mask)
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.6 tdrv011SetBitTiming

NAME

tdrv011SetBitTiming – set bit timing register (transfer rate)

SYNOPSIS

```
TDRV011_STATUS tdrv011SetBitTiming
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    unsigned short      timingValue,
    unsigned int        useThreeSamples
)
```

DESCRIPTION

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

Keep in mind setting the bit timing before changing into the Bus On state.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object to be defined. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

timingValue

This argument contains the new value for the bit timing register 0 (bit 8…15) and bit timing register 1 (bit 0…7).). Possible transfer rates are between 20 Kbit per second and 1 Mbit per second.
The following defines are predefined timing values for the most common transfer rates:

Value	Transfer Rate (max. distance)	
TDRV011_20KBIT	20 Kbit/s	(max. distance: 3.3 km)
TDRV011_50KBIT	50 Kbit/s	(max. distance: 1.3 km)
TDRV011_100KBIT	100 Kbit/s	(max. distance: 620 m)
TDRV011_125KBIT	125 Kbit/s	(max. distance: 530 m)
TDRV011_250KBIT	250 Kbit/s	(max. distance: 270 m)
TDRV011_500KBIT	500 Kbit/s	(max. distance: 130 m)
TDRV011_1MBIT	1 Mbit/s	(max. distance: 40 m)

For other transfer rates please follow the instructions of the Intel 82527 Architectural Overview

useThreeSamples

If this argument is set to *TDRV011_USE_THREE_SAMPLES* the CAN bus is sampled three times per bit time instead of one time.

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.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Set CAN bus bit timing
*/
result = tdrv011SetBitTiming (
                        hdl,
                        2,                  /* channel */
                        TDRV011_100KBIT,    /* 100kbits */
                        0);                 /* single sample point */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.
TDRV011_ERR_INUSE	The channel must be stopped (BUS OFF) state to change the bit timing.

Other returned error codes are system error conditions.

4.2.7 tdrv011Start

NAME

tdrv011Start – set controller online (BUS ON)

SYNOPSIS

```
TDRV011_STATUS tdrv011Start
(
    TDRV011_HANDLE      hdl,
    int                 canChannel
)
```

DESCRIPTION

This function starts setting the specified CAN controller into the BUS ON state. After entering BUS ON the controller is able to receive and transmit messages.

After an abnormal rate of occurrences of errors on the CAN bus, the CAN controller enters the BUS OFF state. This I/O control function resets the init bit in the Control register. The CAN controller begins the bus recovery sequence. The bus recovery sequence 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 CAN controller can communicate over the CAN Bus after driver start-up or a previous BUS OFF error condition this control function must be executed.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel that should be set BUS ON. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Get channel 0 into BUS ON state
*/
result = tdrv011Start ( hdl,
                        0);       /* channel */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.8 tdrv011Stop

NAME

tdrv011Stop – set controller offline (BUS OFF)

SYNOPSIS

```
TDRV011_STATUS tdrv011Stop
(
      TDRV011_HANDLE      hdl,
      int                 canChannel
)
```

DESCRIPTION

This function sets the specified CAN controller into the BUS OFF state. The controller will stop execution of transmitting and receiving messages.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel that should be set BUS OFF. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

EXAMPLE

```
#include "tdrv011api.h"
TDRV011_HANDLE     hdl;
int                result;

/*
** Get channel 0 into BUS OFF state
*/
result = tdrv011Stop (  hdl,
                        0);       /* channel */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.9 tdrv011DefineReceiveMsgObj

NAME

tdrv011DefineReceiveMsgObj – define a message object for receive

SYNOPSIS

```
TDRV011_STATUS tdrv011DefineReceiveMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 msgObjNumber,
    unsigned int        identifier,
    unsigned int        flags,
    int                 rcvMsgQueueNumber
)
```

DESCRIPTION

This function sets up a free controller message object to receive CAN messages with a specified identifier and assigns a queue where the received messages will be buffered.

Before the driver can receive CAN messages it is necessary to define at least one receive message object. If only one receive message object is defined at all, preferably message object 15 should be used because only this message object is double-buffered.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number for which the message object shall be defined. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

msgObjNumber

This argument contains the number of the message object to define. Valid numbers are in the range between 1 and 15 with exception of the number of the default transmit object (usually 1).

identifier

This argument specifies the message identifier that should be received by this message object. Depending on the acceptance filter configuration this initial message identifier value may be changed by other accepted messages with different identifiers. This may cause confusion after changing the acceptance filter masks without redefining the receive message objects.

flags

This argument specifies if the message object shall accept standard or extended CAN message frames. The following flags may be set:

Value	Description
TDRV011_STANDARD_IDENTIFIER	Set if the message object shall accept standard CAN message frames.
TDRV011_EXTENDED_IDENTIFIER	Set if the message object shall accept extended CAN message frames.

rcvMsgQueueNumber

This argument specifies the connected receive message queue. All received messages of the specified message object will be transferred into the specified queue and can read by *tdrv011read()* and *tdrv011readNoWait()* by specifying this *rcvMsgQueueNumber*.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Define message object 15 to receive extended messages with the
** specified identifier
*/
result = tdrv011DefineReceiveMsgObj (
                        hdl,
                        2,            /* channel */
                        15,           /* message object 15 */
                        1234,         /* message identifier */
                        TDRV011_EXTENDED_IDENTIFIER,
                        1);           /* receive queue */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.
TDRV011_ERR_BUSY	The specified message object is already in use.

Other returned error codes are system error conditions.

4.2.10 tdrv011DefineRemoteMsgObj

NAME

tdrv011DefineRemoteMsgObj – define a remote message object

SYNOPSIS

```
TDRV011_STATUS tdrv011DefineRemoteMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 msgObjNumber,
    unsigned int        identifier,
    unsigned int        extMsgFlag,
    int                 length,
    unsigned char       *pData
)
```

DESCRIPTION

This function defines a remote transmission CAN message buffer object. A remote transmission object is similar to normal transmission object, except the CAN message is only transmitted after receipt of a remote frame with the specified identifier.

This type of message object can be used to make process data available for other nodes which can be polled around the CAN bus without any action of the provider node.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object shall be define. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

msgObjNumber

This argument contains the number of the message object to define. Valid numbers are in the range between 1 and 14 with exception of the number of the default transmit object (usually 1). Keep in mind that message object 15 is available only for receive message objects.

identifier

This argument specifies the message identifier for this message object.

extMsgFlag

This argument specifies if the message object shall handle standard or extended CAN message frames. The following flags may be set:

Value	Description
TDRV011_STANDARD_IDENTIFIER	Set if the message object shall handle standard CAN message frames.
TDRV011_EXTENDED_IDENTIFIER	Set if the message object shall handle extended CAN message frames.

length

Specifies the number of valid data bytes stored in pData. The maximum data length is 8.

pData

This argument points to a buffer that contains the message data, which will be send on request. Data[0] contains message Data 0, Data[1] contains message Data 1 and so on.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Define a remote message object that answers to standard identifier 66
*/
result = tdrv011DefineRemoteMsgObj (
                        hdl,
                        0,            /* channel */
                        6,            /* message object 6 */
                        66,           /* message identifier */
                        TDRV011_STANDARD_IDENTIFIER,
                        5,            /* message data length */
                        "Hello");     /* message data */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.
TDRV011_ERR_BUSY	The specified message object is already in use.

Other returned error codes are system error conditions.

4.2.11 tdrv011UpdateRemoteMsgObj

NAME

tdrv011UpdateRemoteMsgObj – update a remote message object

SYNOPSIS

```
TDRV011_STATUS tdrv011UpdateRemoteMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 msgObjNumber,
    int                 length,
    unsigned char       *pData
)
```

DESCRIPTION

This function updates a former defined remote transmission CAN message buffer object. The function updates message data and length.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object shall be updated. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

msgObjNumber

This argument contains the number of the message object to update. Valid numbers are in the range between 1 and 14 with exception of the number of the default transmit object (usually 1). Keep in mind that message object 15 is available only for receive message objects.

length

This argument specifies the number of valid data bytes stored in pData. The maximum data length is 8.

pData

This argument points to a buffer that contains the message data, which will be send on request. Data[0] contains message Data 0, Data[1] contains message Data 1 and so on.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Updates message data of a remote message object
*/
result = tdrv011UpdateRemoteMsgObj (
                        hdl,
                        0,            /* channel */
                        6,            /* message object 6 */
                        8,            /* message data length */
                        "NewData");   /* message data */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.12 tdrv011UpdateReceiveMsgObj

NAME

tdrv011UpdateReceiveMsgObj – update a receive message object (send a remote frame)

SYNOPSIS

```
TDRV011_STATUS tdrv011UpdateReceiveMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 msgObjNumber,
    int                 length
)
```

DESCRIPTION

This function updates a former defined receive CAN message buffer object. When updating a receive message object the controller transmits a remote frame to the CAN bus. Other nodes may answer to the remote frame and send a CAN message which will be received like a normal CAN message.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object to be updated. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

msgObjNumber

This argument contains the number of the message object to update. Valid numbers are in the range between 1 and 15 with exception of the number of the default transmit object (usually 1).

length

This argument specifies the length of the requested data. The TDRV011 inserts the specified length into the remote frame. The information may be used by the remote node.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Send a remote frame from message object 3 (previously defined)
*/
result = tdrv011UpdateReceiveMsgObj ( hdl,
                                      0,        /* channel */
                                      3,        /* message object 3 */
                                      6);       /* request 6 databyte */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	Invalid value specified.

Other returned error codes are system error conditions.

4.2.13 tdrv011ReleaseMsgObj

NAME

tdrv011ReleaseMsgObj – release a message object

SYNOPSIS

```
TDRV011_STATUS tdrv011ReleaseMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 msgObjNumber
)
```

DESCRIPTION

This function releases a previously defined CAN message object. Any CAN bus transactions of the specified message object will be disabled.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number on which the message object shall be released. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

msgObjNumber

This argument contains the number of the message object to release. Valid numbers are in the range between 1 and 15 with exception of the number of the default transmit object (usually 1).

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Release message object 6 (previously defined)
*/
result = tdrv011ReleaseMsgObj (  hdl,
                                 0,        /* channel */
                                 6);       /* message object 6 */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.14 tdrv011FlushReceiveFifo

NAME

tdrv011FlushReceiveFifo – flush receive message queue(s)

SYNOPSIS

```
TDRV011_STATUS tdrv011FlushReceiveFifo
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    int                 rcvQueue
)
```

DESCRIPTION

This function flushes the specified receive message queue.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number the specified receive queue is assigned to. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

rcvQueue

This argument specifies the receive message queue which shall be flushed. Valid queue numbers are 1 up to the configured maximum. If *rcvQueue* is set to 0 all receive queues on the channel are flushed.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;

/*
** Flush receive message queue 1 of channel 0
*/
result = tdrv011FlushReceiveFifo (    hdl,
                                      0,        /* channel */
                                      1);       /* receive queue 1 */
if (result != TDRV011_OK)
{
     /* handle error */
}

…

/*
** Flush all receive message queues of channel 1
*/
result = tdrv011FlushReceiveFifo (    hdl,
                                      1,        /* channel */
                                      0);       /* all receive queues */
if (result != TDRV011_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

4.2.15 tdrv011GetControllerStatus

NAME

tdrv011GetControllerStatus – get controller status

SYNOPSIS

```
TDRV011_STATUS tdrv011GetControllerStatus
(
    TDRV011_HANDLE      hdl,
    int                 canChannel,
    unsigned char       *pCanStatus
)
```

DESCRIPTION

This function reads the current state of the CAN controller status register for diagnostic purpose.

PARAMETERS

hdl

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

canChannel

This argument specifies the CAN channel number where the controller status shall be read from. Channel numbers are starting with 0 for the 1st channel on a TDRV011 device, 1 for the 2nd channel on a TDRV011 device and so on. The last valid channel number depends on the installed module type.

pCanStatus

This argument is a pointer to an unsigned char variable where the status will be stored in. The status represents the value of the controller status register. The content is described in the *Intel 82527 Architectural Overview*.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE     hdl;
int                result;
unsigned char      contStat;

/*
** Read CAN controller status of channel 0
*/
result = tdrv011GetControllerStatus ( hdl,
                                       0,        /* channel */
                                       &contStat);
if (result != TDRV011_OK)
{
     /* handle error */
}
else
{
     printf("CAN-Controller Status: 0x%02X\n", contStat);
}
```

RETURN VALUE

On success, TDRV011_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	Invalid TDRV011 Device Handle specified
TDRV011_ERR_INVAL	A buffer pointer is NULL, or specified value contains an invalid value.

Other returned error codes are system error conditions.

5 Appendix

5.1 Step by Step Initialization

5.1.1 Transmit and Receive Messages

This chapter gives an overview over the necessary steps of starting a CAN channel.

1. Configure bit timing - tdrv011SetBitTiming()
2. Configure Acceptance Masks - tdrv011SetFilter()
3. Define receive message objects – tdrv011DefineReceiveMsgObj()
4. Start CAN controller operation – tdrv011Start()

The channel can now be used to receive and transmit messages

5.1.2 Answer to Remote Frames

This chapter gives an overview over the necessary steps of setting up a transmit remote message object answering on remote frames.

1. Configure bit timing - tdrv011SetBitTiming()
2. Configure Acceptance Masks - tdrv011SetFilter()
3. Define remote message objects – tdrv011DefineRemoteMsgObj()
4. Start CAN controller operation – tdrv011Start()

The channel will now answer to remote frames (with matching identifiers).

5. Update remote objects to provide new data - tdrv011UpdateRemoteMsgObj()

5.1.3 Request Remote Messages

This chapter gives an overview over the necessary steps of starting a CAN channel requesting remote frames and reading the requested data.

1. Configure bit timing - tdrv011SetBitTiming()
2. Configure Acceptance Masks - tdrv011SetFilter()
3. Define receive message object – tdrv011DefineReceiveMsgObj()
4. Start CAN controller operation – tdrv011Start()

The channel will now answer to remote frames (with matching identifiers).

5. Request remote message – tdrv011UpdateReceiveMsgObj()
6. Read received remote message – tdrv011Read()