The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV011-SW-42

VxWorks Device Driver

Extended CAN Bus

Version 5.0.x

User Manual

Issue 5.0.0

November 2017

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
49 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19
ail: info@tews.com www.tews.com

TDRV011-SW-42

VxWorks Device Driver

Extended CAN Bus

Supported Modules:
TPMC316
TPMC816
TPMC901

Issue	Description	Date
1.0.0	First Issue	December 21, 2006
1.0.1	Description: BSP dependent adjustment added	January 25, 2007
1.0.2	Description: BSP dependent adjustment changed	January 29, 2007
1.1.0	BUSOFF function added	November 21, 2008
1.2.0	Transmit Message Object configuration added	April 23, 2009
2.0.0	VxBus, SMP Support and API description added	June 7, 2010
3.0.0	64-Bit Support added	December 21, 2011
4.0.0	Argument length to the API function tdrv011RequestRemoteData() added	August 9, 2012
5.0.0	Basic I/O Functions removed. VxWorks 7 support added. Installation moved to a new manual.	November 1, 2017

Table of Contents

1 Introduction

1.1 Device Driver

The TDRV011-SW-42 VxWorks device driver software allows the operation of the supported PMC conforming to the VxWorks I/O system specification.

The TDRV011-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new VxBus-enabled (GEN1 or GEN2) driver model. The VxBus-enabled driver is recommended for new developments with later VxWorks 6.x releases and mandatory for VxWorks 64-bit and SMP systems.

Both drivers, legacy and VxBus, share the same application programming interface (API) and invoke a mutual exclusion and binary semaphore mechanism to prevent simultaneous requests by multiple tasks from interfering with each other.

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

The TDRV011-SW-42 device driver supports the following features:

- sending and receiving CAN messages
- extended and standard message frames
- acceptance filtering
- message objects
- remote frame requests

The TDRV011-SW-42 supports the modules listed below:

TPMC316-xx	2 Channel Extended CAN	(PMC, Conduction Cooled)
TPMC816-10	2 Channel Extended CAN	(PMC)
TPMC816-11	1 Channel Extended CAN	(PMC)
TPMC901-10	6 Channel Extended CAN	(PMC)
TPMC901-11	4 Channel Extended CAN	(PMC)
TPMC901-12	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 supported devices it is recommended to read the manuals listed below.

TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide
User Manual of the appropriate hardware
Architectural Overview of the Intel 82527 CAN controller

2 API Documentation

2.1 General Functions

2.1.1 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. The first CAN channel is named “/tdrv011/0”, the second channel is named “/tdrv011/1” and so on.
To see which TDRV011 devices are available in the system call the “devs” command from the VxWorks shell. To get detailed information of associated TDRV011 devices the tdrv011Show function (only VxBus driver) can be called from the VxWorks shell.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;

/*
** open file descriptor to a device
*/
hdl = tdrv011Open("/tdrv011/0");
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

Error Code	Description
TDRV011_ERR_INVAL	Illegal device name
TDRV011_ERR_NODEV	Device not found

2.1.2 tdrv011Close

NAME

tdrv011Close – closes a device.

SYNOPSIS

```
TDRV011_STATUS tdrv011Close
(
      TDRV011_HANDLE              hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

This value specifies the file 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

Error Code	Description
TDRV011_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.3 tdrv011GetModuleInfo

NAME

tdrv011GetModuleInfo – get information of the module

SYNOPSIS

```
TDRV011_STATUS tdrv011GetModuleInfo
(
    TDRV011_HANDLE          hdl,
    unsigned int            *pModuleType,
    unsigned int            *pChannelNo,
    TDRV011_PCIINFO_BUF     *pPciInfoBuf
)
```

DESCRIPTION

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

PARAMETERS

hdl

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

pModuleType

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

Value	Description
TDRV011_MODTYPE_TPMC316	Current module is a TPMC316
TDRV011_MODTYPE_TPMC816	Current module is a TPMC816
TDRV011_MODTYPE_TPMC901	Current module is a TPMC901

pChannelNo

This argument is a pointer to an *unsigned int* data buffer, where the local channel number (zero-based) of the device is returned. Possible values are 0 to 5.

pPciInfoBuf

This argument is a pointer to the structure TDRV011_PCIINFO_BUF that receives information of the module PCI header.

```
typedef struct
{
    unsigned short      vendorId;
    unsigned short      deviceId;
    unsigned short      subSystemId;
    unsigned short      subSystemVendorId;
    int                 pciBusNo;
    int                 pciDevNo;
    int                 pciFuncNo;
} TDRV011_PCIINFO_BUF;
```

vendorId

PCI module vendor ID.

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module sub system vendor ID

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE          hdl;
TDRV011_STATUS          result;
TDRV011_PCIINFO_BUF     pciInfoBuf;
unsigned int            moduleType;
unsigned int            channelNo;

/*
** get module PCI information
*/
result = tdrv011GetModuleInfo( hdl, &moduleType, &channelNo, &pciInfoBuf);

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	The specified device handle is invalid
TDRV011_ERR_INVAL	Specified pointer is invalid.

2.2 Device Access Functions

2.2.1 tdrv011Read

NAME

tdrv011Read – read a CAN message

SYNOPSIS

```
TDRV011_STATUS tdrv011Read
(
    TDRV011_HANDLE      hdl,
    int                 flushBeforeRead,
    int                 timeout,
    unsigned int        *pIdentifier,
    int                 *pExtended,
    int                 *pLength,
    unsigned char       *pData
)
```

DESCRIPTION

This function reads a CAN message from the specified channel. If no message is available the call is blocked until a message is received or the request times out.

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

PARAMETERS

hdl

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

flushBeforeRead

If the message FIFO should be flushed before initiating the read sequence this parameter must be set to TRUE, otherwise to FALSE.

timeout

The parameter timeout specifies the timeout interval, in units of milliseconds. A timeout value of 0 means wait indefinitely.

pIdentifier

This parameter is a pointer to an unsigned int value where the received CAN message identifier is stored.

pExtended

This parameter is a pointer to an int value. The value is set to 0 if a standard message frame was received and >0 if an extended message frame was received.

pLength

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned int        msgIdentifier;
int                 msgExtended;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Read a CAN message from the device. Don't flush before reading.
** If no message is available the read request times out after 1000 ms.
*/
result = tdrv011Read(    hdl,
                         FALSE,
                         1000,
                         &msgIdentifier,
                         &msgExtended,
                         &msgLength,
                         &msgData[0]
                         );

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	The specified device handle is invalid.
TDRV011_ERR_TIMEOUT	Read was blocked and the allowed time has elapsed.
TDRV011_ERR_BUSOFF	The controller is in bus OFF state and no message is available in the receive queue. Note, as long as CAN messages are available in the receive queue FIFO, bus OFF conditions were not reported by the read function. This means you can read all CAN messages out of the receive queue FIFO during bus OFF state without an error result.

2.2.2 tdrv011ReadNoWait

NAME

tdrv011ReadNoWait – read a CAN message (non-blocked)

SYNOPSIS

```
TDRV011_STATUS tdrv011ReadNoWait
(
    TDRV011_HANDLE      hdl,
    unsigned int        *pIdentifier,
    int                 *pExtended,
    int                 *pLength,
    unsigned char       *pData
)
```

DESCRIPTION

This function reads a CAN message from the specified channel. If no message is available the call returns immediately.

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

PARAMETERS

hdl

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

pIdentifier

This parameter is a pointer to an unsigned int value where the received CAN message identifier is stored.

pExtended

This parameter is a pointer to an int value. The value is set to 0 if a standard message frame was received and >0 if an extended message frame was received.

pLength

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned int        msgIdentifier;
int                 msgExtended;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Read a CAN message from the device.
*/
result = tdrv011ReadNoWait(   hdl,
                              &msgIdentifier,
                              &msgExtended,
                              &msgLength,
                              &msgData[0]
                              );

if (result != STATUS_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	The specified device handle is invalid.
TDRV011_ERR_TIMEOUT	Read was blocked and the allowed time has elapsed.
TDRV011_ERR_BUSOFF	The controller is in bus OFF state and no message is available in the receive queue. Note, as long as CAN messages are available in the receive queue FIFO, bus OFF conditions were not reported by the read function. This means you can read all CAN messages out of the receive queue FIFO during bus OFF state without an error result.
TDRV011_ERR_NODATA	No received data available.

2.2.3 tdrv011Write

NAME

tdrv011Write – write a CAN message

SYNOPSIS

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

DESCRIPTION

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

By default the write function uses message object 1 for transmit. If this isn't suitable the default transmit message object can be changed by redefining the macro DEFAULT_TXOBJ in tdrv011def.h. Valid message objects are in the range between 1 and 14.

PARAMETERS

hdl

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

timeout

The parameter timeout specifies the timeout interval, in units of milliseconds. A timeout value of 0 means wait indefinitely.

identifier

Contains the message identifier of the CAN message to write.

extended

Set this parameter to TRUE to send an extended message frame or to FALSE to send a standard message frame.

length

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned int        msgIdentifier;
int                 msgExtended;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Write a CAN message
*/
msgIdentifier  = 1234;
msgExtended    = TRUE;
msgLength      = 2;
msgData[0]     = 0xaa;
msgData[1]     = 0xbb;

result = tdrv011Write(   hdl,
                         1000,
                         msgIdentifier,
                         msgExtended,
                         msgLength,
                         &msgData[0]
                         );

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	The specified device handle is invalid.
TDRV011_ERR_TIMEOUT	The allowed time to finish the write request is elapsed.
TDRV011_ERR_BUSOFF	The controller is in bus OFF state and unable to transmit messages.

2.2.4 tdrv011WriteNoWait

NAME

tdrv011WriteNoWait – write a CAN message (non-blocked)

SYNOPSIS

```
TDRV011_STATUS tdrv011WriteNoWait
(
    TDRV011_HANDLE      hdl,
    unsigned int        identifier,
    int                 extended,
    int                 length,
    unsigned char       *pData
)
```

DESCRIPTION

This function writes a messages to the specified device for subsequent transmission on the CAN bus. The call will return immediately after the message was written to the CAN controller or an error occurred.

By default the write function uses message object 1 for transmit. If this isn't suitable the default transmit message object can be changed by redefining the macro DEFAULT_TXOBJ in tdrv011def.h. Valid message object are in the range between 1 and 14.

PARAMETERS

hdl

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

identifier

Contains the message identifier of the CAN message to write.

extended

Set this parameter to TRUE to send an extended message frame or to FALSE to send a standard message frame.

length

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned int        msgIdentifier;
int                 msgExtended;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Write a CAN message to the device.
*/
msgIdentifier  = 1234;
msgExtended    = TRUE;
msgLength      = 2;
msgData[0]     = 0xaa;
msgData[1]     = 0xbb;

result = tdrv011WriteNoWait(  hdl,
                              msgIdentifier,
                              msgExtended,
                              msgLength,
                              &msgData[0]
                              );

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	The specified device handle is invalid.
TDRV011_ERR_TIMEOUT	The device is busy and cannot send the message immediately.
TDRV011_ERR_BUSOFF	The controller is in bus OFF state and unable to transmit messages.

2.2.5 tdrv011SetFilter

NAME

tdrv011SetFilter – set acceptance filter masks

SYNOPSIS

```
TDRV011_STATUS tdrv011SetFilter
(
    TDRV011_HANDLE      hdl,
    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.

PARAMETERS

hdl

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

globalMaskStandard

This parameter 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 parameter 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 parameter 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.

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned short      globalMaskStandard;
unsigned int        globalMaskExtended;
unsigned int        message15Mask;

/*
** Set acceptance filter.
*/
globalMaskStandard  = 0xffff;
globalMaskExtended  = 0xffffffff;
message15Mask       = 0;

result = tdrv011SetFilter(    hdl,
                              globalMaskStandard,
                              globalMaskExtended,
                              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	The specified device handle is invalid.

2.2.6 tdrv011GetFilter

NAME

tdrv011GetFilter – get acceptance filter masks

SYNOPSIS

```
TDRV011_STATUS tdrv011GetFilter
(
        TDRV011_HANDLE          hdl,
        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.

PARAMETERS

hdl

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

pGlobalMaskStandard

This parameter is a pointer to an unsigned short (16bit) value where the content of the Global Mask-Standard Register is stored. The 11 bit identifier appears in bit position 5..15.

pGlobalMaskExtended

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

pMessage15Mask

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

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned short      globalMaskStandard;
unsigned int        globalMaskExtended;
unsigned int        message15Mask;

/*
** Get acceptance filter.
*/

result = tdrv011GetFilter(    hdl,
                              &globalMaskStandard,
                              &globalMaskExtended,
                              &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	The specified device handle is invalid.

2.2.7 tdrv011SetBitTiming

NAME

tdrv011SetBitTiming – set bit timing register

SYNOPSIS

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

DESCRIPTION

This function modifies the bit timing register of the CAN controller to setup a new CAN bus transfer speed.
Keep in mind to setup a valid bit timing before entering the BUSON state.

PARAMETERS

hdl

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

timingValue

This parameter 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 1.0 MBit per second. The file tdrv011api.h contains predefined transfer rates. (For other transfer rates please follow the instructions of the Intel 82527 Architectural Overview).

useThreeSamples

If this parameter is set to TRUE the CAN bus is sampled three times per bit time instead of a single time (FALSE).

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;

/*
** Set CAN bus bit timing
*/

result = tdrv011SetBitTiming(hdl,
                             TDRV011_100KBIT,
                             FALSE
                             );

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	The specified device handle is invalid.

2.2.8 tdrv011DefineReceiveMsgObj

NAME

tdrv011DefineReceiveMsgObj – setup a message object for receive

SYNOPSIS

```
TDRV011_STATUS tdrv011DefineReceiveMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 messageNumber,
    unsigned int        identifier,
    int                 extended
)
```

DESCRIPTION

This function sets up a free controller message object to receive CAN messages with the specified identifier.
Before the driver can receive CAN messages it's 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 this message object is double-buffered.

PARAMETERS

hdl

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

messageNumber

This parameter 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 parameter 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.

extended

Set to TRUE to receive extended message frames or to FALSE to receive standard message frames.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
int                 msgNum;
unsigned int        msgIdentifier;

/*
** Define message object 15 to receive extended messages with the
** specified identifier
*/
msgNum = 15;
msgIdentifier = 1;

result = tdrv011DefineReceiveMsgObj(    hdl,
                                        msgNum,
                                        msgIdentifier,
                                        TRUE
                                        );

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	The specified device handle is invalid.
TDRV011_ERR_BUSY	The specified message object is already in use.
TDRV011_ERR_INVAL	Illegal message object number specified.

2.2.9 tdrv011DefineTransmitMsgObj

NAME

tdrv011DefineTransmitMsgObj – setup a message object for transmit

SYNOPSIS

```
TDRV011_STATUS tdrv011DefineTransmitMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 messageNumber,
    unsigned int        identifier,
    int                 extended,
    int                 length,
    unsigned char       *pData
)
```

DESCRIPTION

This function sets up a free controller message object to transmit CAN messages immediately without waiting for an acknowledge.

PARAMETERS

hdl

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

messageNumber

This parameter contains the number of the message object to define. Valid numbers are in the range between 1 and 14.

identifier

Contains the message identifier of the CAN message to write.

extended

Set this parameter to TRUE to send an extended message frame or to FALSE to send a standard message frame.

length

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
int                 msgNum;
unsigned int        msgIdentifier;
int                 msgExtended;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Define a transmit buffer object to send the CAN message
** immediately
*/
msgNum         = 2;
msgIdentifier  = 1234;
msgExtended    = TRUE;
msgLength      = 2;
msgData[0]     = 0xaa;
msgData[1]     = 0xbb;

result = tdrv011DefineTransmitMsgObj(   hdl,
                                        msgNum,
                                        msgIdentifier,
                                        msgExtended,
                                        msgLength,
                                        &msgData[0]
                                        );

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	The specified device handle is invalid.
TDRV011_ERR_BUSY	The specified message is already in use.
TDRV011_ERR_INVAL	Illegal message object number specified.

2.2.10 tdrv011DefineRemoteMsgObj

NAME

tdrv011DefineRemoteMsgObj – setup a remote transmit buffer message object

SYNOPSIS

```
TDRV011_STATUS tdrv011DefineRemoteMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 messageNumber,
    unsigned int        identifier,
    int                 extended,
    int                 length,
    unsigned char       *pData
)
```

DESCRIPTION

This function sets up a free controller message object as remote transmission buffer.

A remote transmit buffer object is similar to normal transmission object with exception that the CAN message is transmitted only after receiving of a remote frame with the same 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.

The message data remain available for other CAN nodes until this message object is updated with tdrv011UpdateRemoteMsgObj or released with tdrv011ReleaseMsgObj.

PARAMETERS

hdl

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

messageNumber

This parameter contains the number of the message object to define. Valid numbers are in the range between 1 and 14.

identifier

Contains the message identifier of the CAN message to write.

extended

Set this parameter to TRUE to send extended message frames or to FALSE to send standard message frames.

length

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
int                 msgNum;
unsigned int        msgIdentifier;
int                 msgExtended;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Define a remote buffer message object
*/
msgNum          = 4;
msgIdentifier   = 1234;
msgExtended     = TRUE;
msgLength       = 2;
msgData[0]      = 0xaa;
msgData[1]      = 0xbb;

result = tdrv011DefineRemoteMsgObj(     hdl,
                                        msgNum,
                                        msgIdentifier,
                                        msgExtended,
                                        msgLength,
                                        &msgData[0]
                                        );

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	The specified device handle is invalid.
TDRV011_ERR_BUSY	The specified message is already in use.
TDRV011_ERR_INVAL	Illegal message object number specified.

2.2.11 tdrv011UpdateRemoteMsgObj

NAME

tdrv011UpdateRemoteMsgObj – update remote buffer message object data

SYNOPSIS

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

DESCRIPTION

This function writes only new data into the previous allocated remote buffer message object without starting transmission. The data can be requested by other CAN nodes by sending a RTR message with the same identifier.

PARAMETERS

hdl

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

messageNumber

This parameter contains the number of the previous defined remote buffer message object. Valid numbers are in the range between 1 and 14.

length

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

pData

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

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
int                 msgNum;
int                 msgLength;
unsigned char       msgData[TDRV011_MSG_LEN];

/*
** Update a remote buffer message object
*/
msgNum          = 2;
msgLength       = 2;
msgData[0]      = 0x12;
msgData[1]      = 0x34;

result = tdrv011UpdateRemoteMsgObj(     hdl,
                                        msgNum,
                                        msgLength,
                                        &msgData[0]
                                        );

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	The specified device handle is invalid.
TDRV011_ERR_BUSY	Transmission is pending.
TDRV011_ERR_INVAL	Illegal message object number specified.
TDRV011_ERR_NOMSG	Message object is not allocated

2.2.12 tdrv011RequestRemoteData

NAME

tdrv011RequestRemoteData – request a remote CAN message

SYNOPSIS

```
TDRV011_STATUS tdrv011RequestRemoteData
(
        TDRV011_HANDLE          hdl,
        int                     messageNumber,
        int                     length
)
```

DESCRIPTION

This function transmits a CAN message with the RTR bit set to request a remote message from other CAN nodes. The requesting receive message object must be previously defined with tdrv011DefineReceiveMsgObj.

PARAMETERS

hdl

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

messageNumber

This parameter contains the number of the previous defined receive message object. Valid numbers are in the range between 1 and 15.

length

Contains the number of requested data bytes (1...8).

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
int                 msgNum;
int                 msgLength;

/*
** Request a remote CAN message via message object 2
** with a length of 8 bytes
*/
msgNum          = 2;
msgLength       = 8;

result = tdrv011RequestRemoteData( hdl,
                                   msgNum,
                                   msgLength
                                   );

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	The specified device handle is invalid.
TDRV011_ERR_BUSY	Transmission is pending.
TDRV011_ERR_INVAL	Illegal message object number specified.
TDRV011_ERR_NOMSG	Message object is not allocated

2.2.13 tdrv011ReleaseMsgObj

NAME

tdrv011ReleaseMsgObj – release a previous defined message object

SYNOPSIS

```
TDRV011_STATUS tdrv011ReleaseMsgObj
(
    TDRV011_HANDLE      hdl,
    int                 messageNumber
)
```

DESCRIPTION

This function marks the specified message object invalid and stops any transaction with this message object.

PARAMETERS

hdl

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

messageNumber

This parameter contains the number of the previous defined message object. Valid numbers are in the range between 1 and 15.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
int                 msgNum;

/*
** Release a previous defined message object
*/
msgNum          = 2;

result = tdrv011ReleaseMsgObj(     hdl,
                                   msgNum
                                   );

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	The specified device handle is invalid.
TDRV011_ERR_INVAL	Illegal message object number specified.

2.2.14 tdrv011Start

NAME

tdrv011Start – set the CAN controller online

SYNOPSIS

```
TDRV011_STATUS tdrv011Start
(
        TDRV011_HANLDE              hdl
)
```

DESCRIPTION

This function sets the CAN controller associated with the specified device online to enter the BUSON state.
After an abnormal rate of occurrences of errors on the CAN bus, the CAN controller enters the BUSOFF 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 BUSOFF state is exited.

Before the CAN controller can communicate over the CAN after driver start-up or a previous BUSOFF error condition this control function must be executed.

PARAMETERS

hdl

This parameter 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;

/*
** Enter the BUSON state
*/

result = tdrv011Start( hdl );

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	The specified device handle is invalid.
TDRV011_ERR_BUSOFF	Unable to enter the BUSON state, the CAN controller is still offline and unable to communicate over the CAN bus.

2.2.15 tdrv011Stop

NAME

tdrv011Stop – set the CAN controller offline

SYNOPSIS

```
TDRV011_STATUS tdrv011Stop
(
      TDRV011_HANDLE              hdl
)
```

DESCRIPTION

This function sets the specified CAN controller associated with the specified device into the bus OFF state. After execution of this control function the CAN controller is completely removed from the CAN bus and cannot communicate until the control function tdrv011Start() is executed.

PARAMETERS

hdl

This parameter 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;

/*
** Enter the BUSOFF state
*/

result = tdrv011Stop( hdl );

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	The specified device handle is invalid.

2.2.16 tdrv011FlushReceiveFifo

NAME

tdrv011FlushReceiveFifo – discard all messages in the receive FIFO

SYNOPSIS

```
TDRV011_STATUS tdrv011FlushReceiveFifo
(
      TDRV011_HANDLE              hdl
)
```

DESCRIPTION

This function flushes the CAN message receive FIFO. All messages will be discarded.

PARAMETERS

hdl

This parameter 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;

/*
** Flush the CAN message receive FIFO
*/

result = tdrv011FlushReceiveFifo( hdl );

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	The specified device handle is invalid.

2.2.17 tdrv011GetControllerStatus

NAME

tdrv011GetControllerStatus – get contents of the CAN controller status register

SYNOPSIS

```
TDRV011_STATUS tdrv011GetControllerStatus
(
    TDRV011_HANDLE      hdl,
    unsigned int        *pControllerStatus
)
```

DESCRIPTION

This function returns the content of the CAN controller status register for diagnostic purposes.

PARAMETERS

hdl

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

pControllerStatus

This parameter is a pointer to an unsigned int value which returns the contents of the CAN controller status register.

EXAMPLE

```
#include "tdrv011api.h"

TDRV011_HANDLE      hdl;
TDRV011_STATUS      result;
unsigned int        controllerStatus

/*
** Read the CAN controller Status register
*/
result = tdrv011GetControllerStatus(    hdl,
                                        &controllerStatus
                                        );

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	The specified device handle is invalid.

3 Debugging and Diagnostic

Especially the TDRV011 VxBus device driver provides functions and debugging statements to display versatile information of the driver installation and status on the debugging console.

By default the TDRV011 show routine is included in the driver and can be called from the VxWorks shell. If this function is not needed or program space is rare the function can be removed from the code by un-defining the macro INCLUDE_TDRV011_SHOW in tdrv011drv.c

The tdrv011Show function displays detailed information about probed modules, assignment of devices respective device names to probed TRDV011 modules and device statistics.

If TDRV011 modules were probed but no devices were created it may helpful to enable debugging code inside the driver code by defining the macro TDRV011_DEBUG in tdrv011drv.c. Certain debug information can be selected by assigning one or more (logical OR) TDRV_DBG_xxx values to the variable tdrv011Debug.

```
-> tdrv011Show
Probed Modules:
    [0] TPMC316: Bus=4, Dev=1, DevId=0x013c, VenId=0x1498, Init=OK, vxDev=0xffff8000000fce10

Associated Devices:
    [0] TPMC316: /tdrv011/0 /tdrv011/1

Device Statistics:
    /tdrv011/0:
        open count = 0
        interrupt count = 2
        bus off count = 0
        receive count = 1
        transmit count = 1
        object overrun = 0
        fifo overrun = 0
        timing value = 0x14
        global standard mask = 0xffff
        global extended mask = 0xffffffff
        local message 15 mask = 0x0
    /tdrv011/1:
        open count = 0
        interrupt count = 2
        bus off count = 0
        receive count = 1
        transmit count = 1
        object overrun = 0
        fifo overrun = 0
        timing value = 0x14
        global standard mask = 0xffff
        global extended mask = 0xffffffff
        local message 15 mask = 0x0
```