The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV011-SW-65

Windows 2000/XP Device Driver

Extended CAN

Version 1.0.x

User Manual

Issue 1.0.2

February 2010

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

TDRV011-SW-65

Windows 2000/XP 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 hint for IOCTL_TDRV011_SETFILTER and IOCTL_TDRV011_DEF_RX_BUF	February 11, 2010

Table of Contents

1 Introduction

The TDRV011-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TDRV011 device family on an Intel or Intel-compatible x86 Windows 2000, Windows XP, Windows XP embedded operating system.

The standard file and device (I/O) functions (CreateFile, CloseHandle, and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

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

- Transmission and receive of Standard and Extended Identifiers
- Up to 15 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)	(Conduction Cooled PMC)
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.

User Manual of the used TDRV011 device (e.g. TPMC816 User Manual)

Engineering Manual of the used TDRV011 device (e.g. TPMC816 Engineering Manual)

2 Installation

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

tdrv011.sys	Windows 2000/XP driver binary
tdrv011.h	Header file with IOCTL code definitions
tdrv011.inf	Windows 2000/XP installation script
EmbeddedIoDeviceClass.dll	Windows WDM device class library
TDRV011-SW-65-1.0.3.pdf	This document
example/tdrv011exa.c	Example application
Release.txt	Release information
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV011 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV011 card(s) and boot-up your system, Windows 2000 / XP setup will show a "***New hardware found***" dialog box.

(1) The "***Upgrade Device Driver Wizard***" dialog box will appear on your screen.
Click "***Next***" button to continue.

(2) In the following dialog box, choose "***Search for a suitable driver for my device***".
Click "***Next***" button to continue.

(3) In Drive A, insert the TDRV011 driver disk; select "***Disk Drive***" in the dialog box.
Click "***Next***" button to continue.

(4) Now the driver wizard should find a suitable device driver on the diskette.
Click "***Next***" button to continue.

(5) Complete the upgrade device driver and click "***Finish***" to take all the changes effect.

After successful installation the TDRV011 device driver will start immediately and creates devices (TDRV011_1, TDRV011_2 ...) for all recognized TDRV011 modules.

2.1.2 Confirming Windows 2000 / XP Installation

To confirm that the driver has been properly loaded in Windows 2000 / XP, perform the following steps:

(1) From Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***".

(2) Click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.

(3) Click the "**+**" in front of "***Embedded I/O***".
The driver "***TEWS TECHNOLOGIES - TDRV011 (Multi Channel Extended CAN)***" should appear.

3 Driver Configuration

3.1 Receive Queue Configuration

After Installation of the TDRV011 Device Driver the number of receive queues and the size of the corresponding FIFO is set to their default values.

Default values are:

Number of Receive Queues	Size of Receive Queue FIFO
2	100

If the default values are not suitable the configuration can be changed by modifying the registry, for instance with regedt32.

To change the number of receive queues the following value must be modified.

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TDRV011\NumRxQueues`

Valid values are in range between 1...14

To change the size of the receive FIFO the following value must be modified.

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TDRV011\FifoSize`

The size value must be greater than 2

4 TDRV011 Device Driver Programming

All of these standard Win32 functions are described in detail in the Windows Platform SDK Documentation (Windows base services / Hardware / Device Input and Output).

For details refer to the Win32 Programmers Reference of your used programming tools (C++, Visual Basic etc.)

4.1 TDRV011 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TDRV011 device driver. Only the required parameters are described in detail.

4.1.1 Opening a TDRV011 Device

Before you can perform any I/O the TDRV011 device must be opened by invoking the CreateFile function. CreateFile returns a handle that can be used to access the TDRV011 device.

```
HANDLE CreateFile(
      LPCTSTR   lpFileName,
      DWORD     dwDesiredAccess,
      DWORD     dwShareMode,
      LPSECURITY_ATTRIBUTES  lpSecurityAttributes,
      DWORD     dwCreationDistribution,
      DWORD     dwFlagsAndAttributes,
      HANDLE    hTemplateFile
)
```

Parameters

lpFileName

Points to a null-terminated string, which specifies the name of the TDRV011 to open. The lpFileName string should be of the form \\.\TDRV011_x to open the device x. The ending x is a one-based number. The first device found by the driver is \\.\TDRV011_1, the second \\.\TDRV011_2 and so on.

dwDesiredAccess

Specifies the type of access to the TDRV011. For the TDRV011 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

dwShareMode

Set of bit flags that specify how the object can be shared. Set to 0.

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TDRV011 devices.

dwCreationDistribution

Specifies which action to take on files that exist, and which action to take when files do not exist. TDRV011 devices must be always opened **OPEN_EXISTING**.

dwFlagsAndAttributes

Specifies the file attributes and flags for the file. This value must be set to 0 (no overlapped I/O).

hTemplateFile

This value must be NULL for TDRV011 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TDRV011 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call *GetLastError*.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TDRV011_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TDRV011 device always open existing
     0,                 // no overlapped I/O
     NULL
);

if (hDevice == INVALID_HANDLE_VALUE) {
     ErrorHandler( "Could not open device" ); // process error
}
```

See Also

CloseHandle(), Win32 documentation CreateFile()

4.1.2 Closing a TDRV011 Device

The CloseHandle function closes an open TDRV011 handle.

BOOL CloseHandle(
HANDLE *hDevice*
)

Parameters

hDevice

Identifies an open TDRV011 handle.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

Example

```
HANDLE hDevice;
if( !CloseHandle( hDevice ) ) {
     ErrorHandler("Could not close device" ); // process error
}
```

See Also

CreateFile (), Win32 documentation CloseHandle ()

4.1.3 TDRV011 Device I/O Control Functions

The DeviceIoControl function sends a control code directly to a specified device driver, causing the corresponding device to perform the specified operation.

```
BOOL DeviceIoControl(
    HANDLE      hDevice,
    DWORD       dwIoControlCode,
    LPVOID      lpInBuffer,
    DWORD       nInBufferSize,
    LPVOID      lpOutBuffer,
    DWORD       nOutBufferSize,
    LPDWORD     lpBytesReturned,
    LPOVERLAPPED  lpOverlapped
)
```

Parameters

hDevice

Handle to the TDRV011 that is to perform the operation.

dwIoControlCode

Specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in tdrv011.h:

Value	Meaning
IOCTL_TDRV011_READ	Read a CAN message from the specified receive queue
IOCTL_TDRV011_READ_NOWAIT	Same as *IOCTL_TDRV011_READ* but do not wait if no CAN message is available.
IOCTL_TDRV011_WRITE	Transmit a CAN message
IOCTL_TDRV011_BITTIMING	Setup new bit timing
IOCTL_TDRV011_SETFILTER	Setup acceptance filter masks
IOCTL_TDRV011_GETFILTER	Get the current acceptance filter masks
IOCTL_TDRV011_BUSON	Enter the bus on state
IOCTL_TDRV011_BUSOFF	Enter the bus off state
IOCTL_TDRV011_FLUSH	Flush one or all receive FIFO's
IOCTL_TDRV011_CAN_STATUS	Returns the contents of the CAN controller status register
IOCTL_TDRV011_DEF_RX_BUF	Define a receive buffer message object
IOCTL_TDRV011_DEF_RMT_BUF	Define a remote transmit buffer message object
IOCTL_TDRV011_UPDATE_BUF	Update a previous defined remote or receive buffer message object
IOCTL_TDRV011_RELEASE_BUF	Release an allocated message buffer object
IOCTL_TDRV011_CONFIG	Setup some device driver specific parameters

See behind for more detailed information on each control code.

lpInBuffer

Pointer to a buffer that contains the data required to perform the operation.

nInBufferSize

Specifies the size, in bytes, of the buffer pointed to by lpInBuffer.

lpOutBuffer

Pointer to a buffer that receives the operation's output data.

nOutBufferSize

Specifies the size, in bytes, of the buffer pointed to by lpOutBuffer.

lpBytesReturned

Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by lpOutBuffer. A valid pointer is required.

lpOverlapped

Pointer to an Overlapped structure. This value must be set to NULL (no overlapped I/O).

To use these TDRV011 specific control codes the header file tdrv011.h must be included.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

Please note that the TDRV011 driver returns always standard Win32 error codes on failure. Please refer to the Windows Platform SDK Documentation for a detailed description of returned error codes.

See Also

Win32 documentation DeviceIoControl ()

4.1.3.1 IOCTL_TDRV011_READ, IOCTL_TDRV011_READ_NOWAIT

This TDRV011 control functions read a CAN message from the specified receive queue of the specified CAN channel. A pointer to the callers message buffer (*TDRV011_MSG_BUF*) is passed by the parameters *lpInBuffer* and *lpOutBuffer* to the driver.

Before execution of this control function the channel and receive queue number in the message buffer (*lpInBuffer*) must be set by the application. After successful execution the message buffer (*lpOutBuffer*) receives the CAN message.

The control function *IOCTL_TDRV011_READ_NOWAIT* returns immediately with the error code *ERROR_NO_DATA* if no data are available.

```
typedef struct {
    USHORT      Channel;
    USHORT      RxQueueNum;
    ULONG       Identifier
    USHORT      Extended;
    USHORT      MsgLen
    UCHAR       Data[8];
    UCHAR       Status;
} TDRV011_MSG_BUF, *PTDRV011_MSG_BUF;
```

Channel

Specifies the CAN channel number from which the data will be read. 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.

RxQueueNum

Specifies the receive queue number from which the data will be read. Valid receive queue numbers are in range between 1 and n. In which n depends on the driver registry configuration parameter NumRxQueues (see also 3.1).

Identifier

Receives the message identifier of the read CAN message.

Extended

Receives TRUE for extended CAN messages.

MsgLen

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

Data[8]

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

Status

Receives 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 was 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 were 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 were overwritten in the FIFO between the interrupt service routine and post-processing in the driver, on lower system priority levels.

Example

```
#include "tdrv011.h"

HANDLE   hDevice;
BOOLEAN success;
ULONG NumBytes;
TDRV011_MSG_BUF  MsgBuf;


MsgBuf.Channel     = 0;
MsgBuf.RxQueueNum  = 1;

success = DeviceIoControl (
     hDevice,                // TDRV011 handle
     IOCTL_TDRV011_READ,     // control code
     &MsgBuf,                // buffer with control information
     sizeof(MsgBuf),
     &MsgBuf,                // buffer which receives the msg
     sizeof(MsgBuf),
     &NumBytes,              // number of bytes transferred
     NULL
);
```

```
if( success ) {
     // Process data
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	The channel number or receive queue number are out of range.
ERROR_CONNECTION_REFUSED	The CAN controller is in BUS OFF state.
ERROR_NO_DATA	No CAN message available for read. Only relevant for IOCTL_TDRV011_READ_NOWAIT ioctl command.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.2 IOCTL_TDRV011_WRITE

This TDRV011 control function writes a CAN message to the specified CAN channel. A pointer to the callers message buffer (*TDRV011_MSG_BUF*) is passed by the parameter *lpInBuffer* to the driver.

Keep in mind to configure the default transmit message object with the control function *IOCTL_TDRV011_CONFIG* before.

```
typedef struct {
    USHORT      Channel;
    USHORT      RxQueueNum;
    ULONG       Identifier;
    USHORT      Extended;
    USHORT      MsgLen;
    UCHAR       Data[8];
    UCHAR       Status;
} TDRV011_MSG_BUF, *PTDRV011_MSG_BUF;
```

Channel

Specifies the CAN channel number from which the data will be read. 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..

RxQueueNum

Unused for this control function. Can be 0.

Identifier

Contains the message identifier of the CAN message to write.

Extended

Contains TRUE for extended CAN messages.

MsgLen

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

Data[8]

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

Status

Unused for this control function. Can be 0.

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_MSG_BUF     MsgBuf;

MsgBuf.Channel      = 0;
MsgBuf.Identifier   = 1234;
MsgBuf.Extended     = TRUE;
MsgBuf.MsgLen       = 2
MsgBuf.Data[0]      = 0xaa;
MsgBuf.Data[1]      = 0xbb;

success = DeviceIoControl (
     hDevice,                // TDRV011 handle
     IOCTL_TDRV011_WRITE,    // control code
     &MsgBuf,                // output buffer
     sizeof(MsgBuf),
     NULL,
     0,
     &NumBytes,              // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	The channel number or receive queue number is out of range.
ERROR_CONNECTION_REFUSED	The CAN controller is in BUS OFF state.
ERROR_REQ_NOT_ACCEP	No transmission object defined (see also IOCTL_TDRV011_CONFIG).

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.3 IOCTL_TDRV011_BITTIMING

This TDRV011 control function modifies the bit timing register of the CAN controller to setup a new CAN bus transfer speed. A pointer to the callers parameter buffer (*TDRV011_BITTIMING*) is passed by the parameter *lpInBuffer* to the driver.

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

```
typedef struct {
    USHORT      Channel;
    USHORT      TimingValue;
    USHORT      ThreeSamples;
} TDRV011_BITTIMING, *PTDRV011_BITTIMING;
```

Channel

Specifies the CAN channel number on which the bit timing to be setup. 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.

Timing Value

This parameter holds the new values for the bit timing register 0 (bit 0...7) and for the bit timing register 1 (bit 8...15). Possible transfer rates are between 20 KBit per second and 1.0 MBit per second. The include file 'tdrv011.h' contains predefined transfer rate symbols (TDRV011_20KBIT ... TDRV011_1_0MBIT).
For other transfer rates please follow the instructions of the Intel 82527 Architectural Overview, which is also part of the engineering kit TDRV011-EK.

ThreeSamples

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

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

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_BITTIMING   BitTimingParam;

…

//
//  Setup 100 kbit transfer rate for CAN channel 1
//
BitTimingParam.Channel      = 1;
BitTimingParam.TimingValue  = TDRV011_100KBIT;
BitTimingParam.ThreeSamples = FALSE;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_BITTIMING,    // control code
     &BitTimingParam,            // parameter buffer
     sizeof(BitTimingParam),
     NULL,
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel number is out of range.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.4 IOCTL_TDRV011_SETFILTER

This TDRV011 control function modifies the acceptance filter masks of the specified CAN Controller.

The acceptance masks allow message objects to receive messages with a larger 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 "1" value means that the incoming bit value "must-match" identically to the corresponding bit in the message identifier.

To be sure that the desired filtering will used, it's recommended to redefine all receive message objects after calling the *IOCTL_TDRV011_SETFILTER* function.

A pointer to the callers parameter buffer (*TDRV011_ACCEPT_MASKS*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    USHORT      GlobalMaskStandard;
    ULONG       GlobalMaskExtended;
    ULONG       Message15Mask;
} TDRV011_ACCEPT_MASKS, *PTDRV011_ACCEPT_MASKS;
```

Channel

Specifies the CAN channel number on which the acceptance filter masks to be modified. 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 parameter specifies 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 mask appears in bit 5...15 of this parameter.

GlobalMaskExtended

This parameter specifies the value for the Global Mask-Extended Register. The Global Mask-Extended Register applies only to messages using the extended CAN identifier. This 29 bit identifier mask appears in bit 3...31 of this parameter.

Message15Mask

This parameter specifies the value for the Message 15 Mask Register. The Message 15 Mask Register is a local mask for message object 15. This 29 bit identifier mask appears in bit 3...31 of this parameter.
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. (See also Intel 82527 Architectural Overview).

Example

```
#include “tdrv011.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TDRV011_ACCEPT_MASKS    AcceptMasksParam;

AcceptMasksParam.Channel = 0;

// Standard identifier bits 0..3 don't care
AcceptMasksParam.GlobalMaskStandard = 0xfe00;

// Extended identifier bits 0..3 don't care
AcceptMasksParam.GlobalMaskExtended = 0xffffff80;

// Message object 15 identifier bits 0..7 don't care
AcceptMasksParam.Message15Mask = 0xfffff800;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_SETFILTER,    // control code
     &AcceptMasksParam,          // parameter buffer
     sizeof(AcceptMasksParam),
     NULL,
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);
if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input or output buffer is too small.
ERROR_INVALID_PARAMETER	The channel number is out of range.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.5 IOCTL_TDRV011_GETFILTER

This TDRV011 control function returns the current acceptance filter masks of the specified CAN Controller.

A pointer to the callers parameter buffer (*TDRV011_ACCEPT_MASKS*) is passed by the parameters *lpInBuffer* and *lpOutBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    USHORT      GlobalMaskStandard;
    ULONG       GlobalMaskExtended;
    ULONG       Message15Mask;
} TDRV011_ACCEPT_MASKS, *PTDRV011_ACCEPT_MASKS;
```

Channel

Specifies the CAN channel number from which the current acceptance filter masks to be read. 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 parameter receives 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 mask appears in bit 5...15 of this parameter.

GlobalMaskExtended

This parameter receives the value for the Global Mask-Extended Register. The Global Mask-Extended Register applies only to messages using the extended CAN identifier. This 29 bit identifier mask appears in bit 3...31 of this parameter.

Message15Mask

This parameter receives the value for the Message 15 Mask Register. The Message 15 Mask Register is a local mask for message object 15. This 29 bit identifier mask appears in bit 3...31 of this parameter.

Example

```
#include “tdrv011.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TDRV011_ACCEPT_MASKS    AcceptMasksParam;

…
```

```
…

AcceptMasksParam.Channel = 0;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_GETFILTER,    // control code
     &AcceptMasksParam,          // input buffer (CAN channel number)
     sizeof(AcceptMasksParam),
     &AcceptMasksParam,          // output buffer with current masks
     sizeof(AcceptMasksParam),
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( success ) {
     printf("\nCurrent acceptance filter masks\n");
     printf("Global Standard Mask Register = 0x%x\n",
          AcceptMasksParam.GlobalMaskStandard);
     printf("Global Extended Mask Register = 0x%x\n",
          AcceptMasksParam.GlobalMaskExtended);
     printf("Message 15 Mask Register      = 0x%x\n",
          AcceptMasksParam.Message15Mask);
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input or output buffer is too small.
ERROR_INVALID_PARAMETER	The channel number is out of range.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.6 IOCTL_TDRV011_BUSON

This TDRV011 control function takes the specified CAN controller into the Bus On state.

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

The only required parameter is the CAN channel number. A pointer to a *USHORT* variable which contains the channel number is passed by the parameters *lpInBuffer* to the driver.

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

Example

```
#include “tdrv011.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
USHORT    Channel;


Channel = 0;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_BUSON,        // control code
     &Channel,                   // pointer the channel number
     sizeof(USHORT),
     NULL,
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel number is out of range.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.7 IOCTL_TDRV011_BUSOFF

This TDRV011 control function takes the specified CAN controller into the BUSOFF state.

After execution of this control function the CAN controller is completely removed from the CAN bus and cannot communicate until the control function *IOCTL_TDRV011_BUSON* was executed.

The only required parameter is the CAN channel number. A pointer to a *USHORT* variable which contains the channel number is passed by the parameters *lpInBuffer* to the driver.

Execute this control function before the last close to the CAN controller channel.

Example

```
#include “tdrv011.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
USHORT    Channel;

Channel = 0;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_BUSOFF,       // control code
     &Channel,                   // pointer the channel number
     sizeof(USHORT),
     NULL,
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel number is out of range.

4.1.3.8 IOCTL_TDRV011_FLUSH

This TDRV011 control function flushes the message FIFO of the specified receive queue(s). A pointer to the callers parameter buffer (*TDRV011_FLUSH*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    USHORT      RxQueueNum;
} TDRV011_FLUSH, *PTDRV011_FLUSH;
```

Channel

Specifies the CAN channel number on which the FIFO's to be flushed. 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.

RxQueueNum

Specifies the receive queue number on which the FIFO's to be flushed. If this parameter is 0 the FIFO's of all receive queues on the specified CAN channel will be flushed, otherwise only the FIFO of the specified receive queue will be flushed.

Example

```
#include "tdrv011.h"

HANDLE        hDevice;
BOOLEAN       success;
ULONG         NumBytes;
TDRV011_FLUSH FlushParam;

//
//  Flush all FIFO's of the specified CAN channel
//
FlushParam.Channel     = 1;
FlushParam.RxQueueNum  = 0;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_FLUSH,        // control code
     &FlushParam,                // parameter buffer
     sizeof(FlushParam),
     NULL,
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);

…
```

…

```
if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel number is out of range.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.9 IOCTL_TDRV011_CAN_STATUS

This TDRV011 control function returns the actual contents of the CAN controller status register for diagnostic purposes.

The only required parameter is the CAN channel number. A pointer to a *USHORT* variable which contains the channel number is passed by the parameter *lpInBuffer* to the driver.

The content of the controller status register will be received in a *UCHAR* variable. A pointer to this variable is passed by the parameter *lpOutBuffer* to the driver.

Example

```
#include “tdrv011.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
USHORT    Channel;
UCHAR     CanStatus;

Channel = 0;

success = DeviceIoControl (
     hDevice,                    // TDRV011 handle
     IOCTL_TDRV011_CAN_STATUS,   // control code
     &Channel,                   // pointer the channel number
     sizeof(USHORT),
     &CanStatus,
     sizeof(UCHAR),
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input or output buffer is too small.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.10 IOCTL_TDRV011_DEF_RX_BUF

This TDRV011 control function defines a CAN message object to receive a single message identifier or a range of message identifiers (see also Acceptance Mask). All CAN messages received by this message object are directed to the associated receive queue and can be read with the standard read function *IOCTL_TDRV011_READ*.

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 buffered.

To be sure that the desired filtering will used, it's recommended to redefine all receive message objects after calling the *IOCTL_TDRV011_SETFILTER* function.

A pointer to the caller's message description (*TDRV011_BUF_DESC*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    SHORT       hannel;
    SHORT       MsgObjNum;
    SHORT       RxQueueNum;
    LONG        Identifier;
    SHORT       Extended;
    SHORT       MsgLen;
    CHAR        Data[8];
} TDRV011_BUF_DESC, *PTDRV011_BUF_DESC;
```

Channel

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.

MsgObjNum

Specifies the number of the message object to be defined. Valid object numbers are in range between 1 and 15.

RxQueueNum

Specifies the associated receive queue for this message object. All CAN messages received by this object are directed to this receive queue. The receive queue number is one based; valid numbers are in range between 1 and n. In which n depends on the driver registry configuration parameter *NumRxQueues*.

It is possible to assign more than one receive message object to a receive queue.

Identifier

Specifies the message identifier for the message object to be defined.

Extended

Set to TRUE for extended CAN messages.

MsgLen

Unused for this control function. Set to 0.

Data[8]

Unused for this control function.

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_BUF_DESC    BufDesc;

//  Define message object 15 to receive the CAN identifier 1234.
BufDesc.Channel    = 0;
BufDesc.MsgObjNum  = 15
BufDesc.RxQueueNum = 1
BufDesc.Identifier = 1234;
BufDesc.Extended   = TRUE;

success = DeviceIoControl (
     hDevice,                          // TDRV011 handle
     IOCTL_TDRV011_DEF_RX_BUF,         // control code
     &BufDesc,                         // message description buffer
     sizeof(BufDesc),
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel number, message object number or receive queue number is out of range.
ERROR_DEVICE_IN_USE	The message object is already occupied.

All other returned error codes are system error conditions.

4.1.3.11 IOCTL_TDRV011_DEF_RMT_BUF

This TDRV011 control function defines a remote transmission CAN message buffer object. A remote transmission object is similar to normal transmission object with exception that the CAN message to be transmitted only after receipt 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 the control function *IOCTL_TDRV011_UPDATE_BUF* or cancelled with *IOCTL_TDRV011_RELEASE_BUF*.

A pointer to the caller's message description (*TDRV011_BUF_DESC*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    USHORT      MsgObjNum;
    USHORT      RxQueueNum;
    ULONG       Identifier;
    USHORT      Extended;
    USHORT      MsgLen;
    UCHAR       Data[8];
} TDRV011_BUF_DESC, *PTDRV011_BUF_DESC;
```

Channel

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.

MsgObjNum

Specifies the number of the message object to be defined. Valid object numbers are in range between 1 and 14.
Keep in mind that message object 15 is available only for receive message objects.

RxQueueNum

Unused for remote transmission message objects. Set to 0.

Identifier

Specifies the message identifier for the message object to be defined.

Extended

Set to TRUE for extended CAN messages.

MsgLen

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

Data[8]

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

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_BUF_DESC    BufDesc;

//  Define message object 2 as remote transmission object
BufDesc.Channel    = 0;
BufDesc.MsgObjNum  = 2
BufDesc.Identifier = 1234;
BufDesc.Extended   = TRUE;
BufDesc.MsgLen     = 1;
BufDesc.Data[0]    = 0x88;

success = DeviceIoControl (
     hDevice,                          // TDRV011 handle
     IOCTL_TDRV011_DEF_RMT_BUF,        // control code
     &BufDesc,                         // message description buffer
     sizeof(BufDesc),
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel number, message object number or message length is out of range.
ERROR_DEVICE_IN_USE	The message object is already occupied.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.12 IOCTL_TDRV011_UPDATE_BUF

This TDRV011 control function updates a previously defined receive or remote transmission message buffer object.

To update a receive message object a remote frame is transmitted over the CAN bus to request new data from a corresponding remote transmission message object on other nodes.

To update a remote transmission object only the message data and message length of the specified message object is changed. No transmission is initiated by this control function.

A pointer to the caller's message description (*TDRV011_BUF_DESC*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    USHORT      MsgObjNum;
    USHORT      RxQueueNum;
    ULONG       Identifier;
    USHORT      Extended;
    USHORT      MsgLen;
    UCHAR       Data[8];
} TDRV011_BUF_DESC, *PTDRV011_BUF_DESC;
```

Channel

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.

MsgObjNum

Specifies the number of the message object to be updated. Valid object numbers are in range between 1 and 14.
Keep in mind that message object 15 is available only for receive message objects.

RxQueueNum

Unused for this control function. Set to 0.

Identifier

Unused for this control function. Set to 0.

Extended

Unused for this control function. Set to 0.

MsgLen

Contains the number of message data bytes (0...8).
This parameter is used only for remote transmission object updates.

Data[8]

This buffer contains up to 8 data bytes. Data[0] contains message Data 0, Data[1] contains message Data 1 and so on.
This parameter is used only for remote transmission object updates.

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_BUF_DESC    BufDesc;


//
//  Update a previously defined receive message object
//
BufDesc.Channel    = 0;
BufDesc.MsgObjNum  = 1

success = DeviceIoControl (
     hDevice,                          // TDRV011 handle
     IOCTL_TDRV011_UPDATE_BUF,         // control code
     &BufDesc,                         // message description buffer
     sizeof(BufDesc),
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}


//
//  Update a previously defined remote transmission message object
//  with new data
//
BufDesc.Channel    = 0;
BufDesc.MsgObjNum  = 2
BufDesc.MsgLen     = 1;
BufDesc.Data[0]    = 0x99;

…
```

…

```
success = DeviceIoControl (
     hDevice,                          // TDRV011 handle
     IOCTL_TDRV011_UPDATE_BUF,         // control code
     &BufDesc,                         // message description buffer
     sizeof(BufDesc),
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel or message object number is out of range or the requested message object isn't allocated and initialized.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.13 IOCTL_TDRV011_RELEASE_BUF

This TDRV011 control function releases a previously defined CAN message object. Any CAN bus transactions of the specified message object become disabled. After releasing the message object can be defined again with *IOCTL_TDRV011_DEF_RX_BUF* and *IOCTL_TDRV011_DEF_RMT_BUF* control functions.

A pointer to the caller's message description (*TDRV011_BUF_DESC*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    USHORT      MsgObjNum;
    USHORT      RxQueueNum;
    ULONG       Identifier;
    USHORT      Extended;
    USHORT      MsgLen;
    UCHAR       Data[8];
} TDRV011_BUF_DESC, *PTDRV011_BUF_DESC;
```

Channel

Specifies the CAN channel number on which the message object to 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.

MsgObjNum

Specifies the number of the message object to be released. Valid object numbers are in range between 1 and 15.

RxQueueNum

Unused for this control function. Set to 0.

Identifier

Unused for this control function. Set to 0.

Extended

Unused for this control function. Set to 0.

MsgLen

Unused for this control function. Set to 0.

Data[8]

Unused for this control function.

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_BUF_DESC    BufDesc;


//
//  Release message object 15
//
BufDesc.Channel    = 0;
BufDesc.MsgObjNum  = 15

success = DeviceIoControl (
     hDevice,                          // TDRV011 handle
     IOCTL_TDRV011_RELEASE_BUF,        // control code
     &BufDesc,                         // message description buffer
     sizeof(BufDesc),
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel or message object number is out of range or the message object isn’t occupied.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.14 IOCTL_TDRV011_CONFIG

This TDRV011 control function configures device specific parameters. It's necessary to execute this control function before any other control function.

A pointer to the callers parameter buffer (*TDRV011_CONFIG*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    USHORT      Channel;
    ULONG       DeviceWriteTimeout;
    ULONG       DeviceReadTimeout;
    USHORT      TxMsgObjNum;
} TDRV011_CONFIG, *PTDRV011_CONFIG;
```

Channel

Specifies the CAN channel number to be configured. 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.

DeviceWriteTimeout

Specifies the timeout in seconds for all following *IOCTL_TDRV011_WRITE* control functions.

DeviceReadTimeout

Specifies the timeout in seconds for all following *IOCTL_TDRV011_READ* control functions.

TxMsgObjNum

Specifies the number of the message object which is used to transmit CAN messages with the *IOCTL_TDRV011_WRITE* control function. Valid transmission object number are in range between 1...14.

As long as the default transmission object is unknown no messages can be sending with the *IOCTL_TDRV011_WRITE* control function.

Example

```
#include “tdrv011.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV011_CONFIG      ConfigParam;

…

//  Configure CAN channel 1
ConfigParam.Channel               = 1;
ConfigParam.DeviceWriteTimeout    = 2;
ConfigParam.DeviceReadTimeout     = 5;
ConfigParam.TxMsgObjNum           = 1;

success = DeviceIoControl (
     hDevice,                          // TDRV011 handle
     IOCTL_TDRV011_CONFIG,             // control code
     &ConfigParam,                     // pointer the parameter buffer
     sizeof(ConfigParam),
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the input buffer is too small.
ERROR_INVALID_PARAMETER	The channel or message object number is out of range or the requested message object is already occupied as receive or remote buffer object.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.2 Step by Step Driver Initialization

The following code example illustrates all necessary steps to initialize a CAN device for communication.

CAN channel will be initialized to communicate with a bus speed of 100 kbit. The default receive object (message object 15) will accept all extended identifiers. Message object 1 is used for transmission.

(1) Configure CAN channel

```
ConfigParam.Channel              = 0;
ConfigParam.DeviceWriteTimeout   = 2;
ConfigParam.DeviceReadTimeout    = 5;
ConfigParam.TxMsgObjNum          = 1;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV011_CONFIG,
     &ConfigParam,
     sizeof(ConfigParam),
     NULL,
     0,
     &NumBytes,
     NULL
);
```

(2) Setup CAN bus bit timing

```
BitTimingParam.Channel      = 0;
BitTimingParam.TimingValue  = TDRV011_100KBIT;
BitTimingParam.ThreeSamples = FALSE;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV011_BITTIMING,
     &BitTimingParam,
     sizeof(BitTimingParam),
     NULL,
     0,
     &NumBytes,
     NULL
);
```

(3) Setup acceptance filter masks

```
AcceptMasksParam.Channel              = 0;
AcceptMasksParam.GlobalMaskStandard   = 0;
AcceptMasksParam.GlobalMaskExtended   = 0;
AcceptMasksParam.Message15Mask        = 0;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV011_SETFILTER,
     &AcceptMasksParam,
     sizeof(AcceptMasksParam),
     NULL,
     0,
     &NumBytes,
     NULL
);
```

(4) Define receive message object

```
BufDesc.Channel    = 0;
BufDesc.MsgObjNum  = 15;
BufDesc.RxQueueNum = 1;
BufDesc.Identifier = 0;
BufDesc.Extended   = TRUE;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV011_DEF_RX_BUF,
     &BufDesc,
     sizeof(BufDesc),
     NULL,
     0,
     &NumBytes,
     NULL
);
```

(5) Enter Bus On State

```
Channel = 0;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV011_BUSON,
     &Channel,
     sizeof(USHORT),
     NULL,
     0,
     &NumBytes,
     NULL
);
```