The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV002-SW-65

Windows Device Driver

Multiple Channel Serial Interface

Version 2.0.x

User Manual

Version 2.0.2

November 2011

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

TDRV002-SW-65

Windows Device Driver

Multiple Channel Serial Interface

Supported Modules:
- TPMC371
- TPMC372
- TPMC375
- TPMC376
- TPMC377
- TPMC460
- TPMC461
- TPMC462
- TPMC463
- TPMC465
- TPMC466
- TPMC467
- TPMC470
- TCP460
- TCP461
- TCP462
- TCP463
- TCP465
- TCP466
- TCP467
- TCP469
- TCP470

This document contains information, which is proprietary to TEWS TECHNOLOGIES GmbH. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES GmbH reserves the right to change the product described in this document at any time without notice.

TEWS TECHNOLOGIES GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2004-2011 by TEWS TECHNOLOGIES GmbH

Issue	Description	Date
1.0.0	First Issue	December 2, 2004
1.1.0	File list extended	April 5, 2005
1.2.0	New modules, support of programmable transceivers	May 15, 2006
1.2.1	File list extended	August 25, 2006
1.2.2	Files moved to subdirectory	June 23, 2008
2.0.0	Support for Windows 7 added	March 9, 2011
2.0.1	Default Configuration in Property Page Chapter 'Known Problems' added	March 29, 2011
2.0.2	Chapter 'Known Problems' modified	November 17, 2011

Table of Contents

1 Introduction

The TDRV002-SW-65 Windows device driver is a kernel mode driver which allows the operation of the supported hardware modules on an Intel or Intel-compatible Windows operating system. Supported Windows versions are:

- Windows 2000
- Windows XP
- Windows XP Embedded
- Windows 7 (32bit and 64bit)

The standard file input and output (I/O) functions (CreateFile, CloseHandle, ReadFile, ReadFileEx, WriteFile, WriteFileEx and DeviceIoControl) provide the basic interface for opening and closing a communications resource handle and for performing read and write operations.

The TDRV002 device driver is fully compatible to the standard Windows serial device driver (*serial.sys*).

The TDRV002-SW-65 device driver supports the modules listed below:

TPMC371	8 Channel Serial Interface	PMC Conduction Cooled
TPMC372	4 Channel Serial Interface	PMC Conduction Cooled
TPMC375	8 Channel Serial Interface	PMC Conduction Cooled
TPMC376	4 Channel Serial Interface	PMC Conduction Cooled
TPMC377	4 Channel Isolated Serial Interface	PMC Conduction Cooled
TPMC460	16 Channel Serial Interface	PMC
TPMC461	8 Channel Serial Interface	PMC
TPMC462	4 Channel Serial Interface	PMC
TPMC463	4 Channel Serial Interface	PMC
TPMC465	8 Channel Serial Interface	PMC
TPMC466	4 Channel Serial Interface	PMC
TPMC467	4 Channel Serial Interface	PMC
TPMC470	4 Channel Isolated Serial Interface	PMC
TCP460	16 Channel Serial Interface	CompactPCI
TCP461	8 Channel Serial Interface	CompactPCI
TCP462	4 Channel Serial Interface	CompactPCI
TCP463	4 Channel Serial Interface	CompactPCI
TCP465	8 Channel Serial Interface	CompactPCI
TCP466	4 Channel Serial Interface	CompactPCI
TCP467	4 Channel Serial Interface	CompactPCI
TCP469	8 Channel Isolated Serial Interface	CompactPCI
TCP470	4 Channel Isolated Serial Interface	CompactPCI

2 Installation

Following files are located in directory TDRV002-SW-65 on the distribution media:

i386\	Directory containing driver files for 32bit Windows versions
amd64\	Directory containing driver files for 64bit Windows versions
installer_32bit.exe	Installation tool for 32bit systems (Windows XP or later)
installer_64bit.exe	Installation tool for 64bit systems (Windows XP or later)
<drivername>amd64.cat	Driver CAT-File (64-bit)
<drivername>i386.cat	Driver CAT-File (32-bit)
<drivername>.inf	Windows installation script
tdrv002.h	Header file with IOCTL codes and structure definitions
example\tdrv002exa.c	Example application
TDRV002-SW-65-2.0.2.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000

This section describes how to install the TDRV002 Device Driver on a Windows 2000 operating system.

After installing the hardware module(s) and boot-up your system, Windows 2000 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. Insert the TDRV002-SW-65 driver disk; select the matching drive of the distribution media in the dialog box.
   Click "***Next***" button to continue.
4. Now the driver wizard should find a suitable device driver on the distribution media.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to make all the changes take effect.
   The driver will create the TDRV002 devices.

Now the bus driver for TEWS *TECHNOLOGIES* serial modules is installed. The system will now ask for the serial device driver and Windows 2000 setup will show a "***New hardware found***" dialog box again.

6. The "***Upgrade Device Driver Wizard***" dialog box will appear on your screen.
   Click "***Next***" button to continue.
7. In the following dialog box, choose "***Search for a suitable driver for my device***".
   Click "***Next***" button to continue.
8. Insert the TDRV002-SW-65 driver disk; select the matching drive of the distribution media in the dialog box.
   Click "***Next***" button to continue.
9. Now the driver wizard should find a suitable device driver on the distribution media.
   Click "***Next***" button to continue.
10. Complete the upgrade device driver and click "***Finish***" to make all the changes take effect.

After successful installation, the TDRV002 device driver will start immediately and create devices (TDRV002_1, TDRV002_2 ...) for all recognized modules supported by the TDRV002-SW-65 device driver.

It may be necessary to restart the system after installation.

2.1.2 Windows XP / Windows 7

This section describes how to install the TDRV002-SW-65 Device Driver on a Windows XP (32bit) or Windows 7 (32bit or 64bit) operating system.

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

Copy needed files (tdrv002.h) to desired target directory.

After successful installation a device is created for each channel found (TDRV002_1, TDRV002_2 ...).

2.2 Confirming Driver Installation

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

1. Open the Windows Device Manager:
   a. For Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***" and click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
   b. For Windows 7, open the "***Control Panel***" from "***My Computer***" and then click the "***Device Manager***" entry.
2. Click the "**+**" in front of "***Embedded I/O***".
   The enumerator device
   "***TEWS TECHNOLOGIES - TDRV002 Family Serial Port Enumerator)***" should appear.
3. Click the "**+**" in front of "***Ports (COM & LPT)***".
   The serial port devices
   "***TEWS TECHNOLOGIES - TDRV002 Family Serial Port Device(…) (COMxx)***" should appear.

3 Default Configuration

The default configuration of the port can be modified by using the property page of the port device.

The property page can be opened from the device manager. A right-click to the port device will open a menu where ‘Properties’ can be selected. The property page will open. The tab ‘Port Settings’ will show the default settings of the port.

All settings will only be used on device startup. Therefore it is necessary to restart the device after modifying any of the settings below. (Restart the device using the device manager, or simply restart the system)

3.1 Basic Port Settings

Using this page the basic settings of the port can be changed. Basic settings are:

- ‘Bits per second’	baud rate	
- ‘Data bits’	number of data	(5, 6, 7, 8)
- ‘Parity’	parity mode	(None, Even, Odd, Space, Mark)
- ‘Stop bits’	number of stopbits	(1, 1.5, 2)
- ‘Flow control’	flow control mode	(None, Xon/Xoff, Hardware)

3.2 Advanced Port Settings

The advanced port settings can be opened by pressing the ’Advanced’ Button at the Basic Port Settings page.

This site allows modification of the buffer trigger levels for ‘Receive Buffer’ and ‘Transmit Buffer’. Increasing a value means, that system load is decreased, but the risk of an overrun for receive, or a gap in transmission stream is increased.

The TDRV002 devices are using a 64 byte internal FIFO, but the property page supports just 16 character FIFOs. Therefore the trigger levels are scaled by the driver with a factor of 4, e.g. a trigger setting of 8 means we are using a trigger level of 32 (8*4).

Disabling the FIFOs is not recommended, this will increase the possibility of data loss and will also increases system load.

The site also allows advising COM-Port numbers. This may be useful for applications that only allow the use of some special port numbers.

4 Device Driver Programming

The Microsoft® Win32® application programming interface (API) also includes a set of functions that provide special communication services like reading and setting communication parameter, transmitting immediate characters, setting timeouts and so on.

All of these standard Win32 communication functions are described in detail in the Windows Platform SDK Documentation (Windows base services / Communication).

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

The Windows name of the first port is *Devicetdrv002_0,* of the second port *Device*\ *tdrv002_1* and so on.**

The DOS device name for TDRV002 devices is COM1, COM2, COM3 and so on. If there are other serial devices in the system the prefix starts with a higher number (see Windows name).

The mapping between Windows device names and DOS device names for TDRV002 devices can be retrieved from the 'Advanced Port Settings'.

4.1 TDRV002 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TDRV002 device driver. Only the required parameters are described in detail.

4.1.1 Opening a TDRV002 Device

Before you can perform any I/O, the TDRV002 device must be opened by invoking the CreateFile function. CreateFile returns a handle that can be used to access the TDRV002 device.

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

Parameters

lpFileName

This parameter points to a null-terminated string, which specifies the name of the TDRV002 to open. The *lpFileName* string should be of the form \\.\COM*x* to open the device *x.*

dwDesiredAccess

This parameter specifies the type of access to the TDRV002.
For the TDRV002 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

This argument is a pointer to a security structure. Set to NULL for TDRV002 devices.

dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TDRV002 devices must be always opened OPEN_EXISTING.

dwFlagsAndAttributes

Specifies the file attributes and flags for the file. This value must be set to 0 for TDRV002 devices.

hTemplateFile

This value must be NULL for TDRV002 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TDRV002 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\COM5”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                    // no security attrs
     OPEN_EXISTING,           // TDRV002 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 TDRV002 Device

The CloseHandle function closes an open TDRV002 handle.

BOOL CloseHandle(
HANDLE *hDevice;*
)

Parameters

hDevice

Identifies an open TDRV002 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 TDRV002 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,          // handle to device of interest
    DWORD           dwIoControlCode,  // control code of operation to perform
    LPVOID          lpInBuffer,       // pointer to buffer to supply input data
    DWORD           nInBufferSize,    // size of input buffer
    LPVOID          lpOutBuffer,      // pointer to buffer to receive output data
    DWORD           nOutBufferSize,   // size of output buffer
    LPDWORD         lpBytesReturned,  // pointer to variable to receive output byte count
    LPOVERLAPPED    lpOverlapped      // pointer to overlapped structure for asynchronous
                                      // operation
)
```

Parameters

hDevice

Handle to the TDRV002 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 *tdrv002.h*:

Value	Meaning
IOCTL_TDRV002_CONF_TRANS	Setup programmable interfaces
other	Other functions for serial drivers are supported by this driver. Please refer to the Microsoft documentation for serial drivers.

See below for more detailed information on each control code.

To use these TDRV002 specific control codes, the header file tdrv002.h must be included in the application

lpInBuffer

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

nInBufferSize

Specifies the size of the buffer pointed to by *lpInBuffer*.

lpOutBuffer

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

nOutBufferSize

Specifies the size of the buffer in bytes 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. Overlapped access is not supported.

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*.

See Also

Win32 documentation DeviceIoControl()

4.1.3.1 IOCTL_TDRV002_CONF_TRANS

This function is used for TDRV002 supported modules with programmable I/O interfaces.

The new configuration value is passed in an unsigned char buffer, pointed to by *lpInBuffer*, to the driver. The buffer must be always an unsigned char type. The argument *nInBufferSize* specifies the size (sizeof(UCHAR)) of the configuration buffer.

The configuration value is an ORed value of the following bits. For a description of the bits, please refer to the Hardware User Manual (Channel Setup) of the module.

Bit No.	Name in HW User Manual
0	RS485/RS232#
1	HDPLX
2	RENA
3	RTERM
4	TTERM
5	SLEW LIMIT
6	SHDN
7	Auto RS485 Operation

Example

```
#include <windows.h>
#include <winioctl.h>
#include “tdrv002.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
UCHAR     IntfConfig;

IntfConfig = 0x00;      // RS232

success = DeviceIoControl
(
     hDevice,                     // TDRV002 handle
     IOCTL_TDRV002_CONF_TRANS,    // control code
     &IntfConfig,
     sizeof(IntfConfig),
     NULL,
     0,
     &NumBytes,
     NULL);                       // not overlapped

…
```

```
…

if(success)
{
     printf("Output port successfully written\n");
}
else
{
     ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_PARAMETER	This function is not supported for the module type.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TDRV002 Hardware User Manual

5 Known Problems

5.1 Order of Serial Ports (only Windows 7)

The order of the Serial Ports shown in the Devices Manager may not match channel numbering on the board. Also the advising of COM Port numbers may not match to the local channel numbers, and also not match to the order shown in the device manager.

Fixing COM Port assignment can be done as described in chapter 3.2 Advanced Port Settings. The local channel number is shown as 'Path' by the device properties.

Stopping and restarting devices by the Device Manager or system restarts will not touch the port assignment.

5.2 COM Port Assignment on Higher Port Numbers

If the COM Port assignment does not start with first unused COM Port or the assignment shows gaps in the COM Port assignment, e.g. the four COM ports of a TPMC466 are assigned to COM7 up to COM10, instead of COM3 up to COM6 as expected, this may be caused by problems when uninstalling devices and drivers. This assignment can be corrected in two steps.

1. Check and remove hidden and no more needed COM devices, if any are found. Therefore it may be necessary to enable hidden devices shown in the device manager. This can be enabled by setting the following Environment Variables:

   ```
   Devmgr_show_details=1
   Devmgr_show_nonpresent_devices=1
   ```

2. Use the COM port assignment as described in the 3.2 Advanced Port Settings to assign the correct COM Port name.

5.3 Settings in HyperTerminal

The driver does not support changing settings with HyperTerminal. Other terminal applications will work fine.