The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV002-SW-65

Windows Device Driver

Multiple Channel Serial Interface

Version 2.1.x

User Manual

Version 2.1.3

July 2021

powerBridge
Computer

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

www.powerbridge.de
info@powerbridge.de

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

TDRV002-SW-65

Windows Device Driver

Multiple Channel Serial Interface

Supported Modules:
- TPMC37x
- TPMC46x
- TPMC47x
- TXMC37x
- TXMC46x
- TCP46x
- TCP47x

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
2.1.0	Support of TXMC375 added, Chapter "Default Configuration" modified Chapter "Device and Software De-installation" added	January 14, 2014
2.1.1	Support of TPMC378 added	May 25, 2014
2.1.2	Support of TCP468, TXMC376, TXMC463 and TXMC465 added Table of Supported Modules renewed	January 22, 2016
2.1.3	Support for Windows 10 added. Support for earlier Windows versions removed.	July 1, 2021

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.

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 compatible to the standard Windows serial device driver (*serial.sys*).

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

Module	Serial Interfaces	Programmable Interfaces	FIFO-Size (Bytes)	Isolated	Form Factor	Conduction Cooled
TPMC371	8		64		PMC	●
TPMC372	4		64		PMC	●
TPMC375	8	●	64		PMC	●
TPMC376	4	●	64		PMC	●
TPMC377	4	●	64	●	PMC	●
TPMC378	8		64	●	PMC	●
TPMC460	16		64		PMC	
TPMC461	8		64		PMC	
TPMC462	4		64		PMC	
TPMC463	4		64		PMC	
TPMC465	8	●	64		PMC	
TPMC466	4	●	64		PMC	
TPMC467	4	●	64		PMC	
TPMC470	4	●	64	●	PMC	
TXMC375	8	●	256		XMC	●
TXMC376	4	●	256		XMC	●
TXMC463	4		256		XMC	
TXMC465	8	●	256		XMC	
TCP460	16		64		cPCI	
TCP461	8		64		cPCI	
TCP462	4		64		cPCI	
TCP463	4		64		cPCI	
TCP465	8	●	64		cPCI	
TCP466	4	●	64		cPCI	

Module	Serial Interfaces	Programmable Interfaces	FIFO-Size (Bytes)	Isolated	Form Factor	Conduction Cooled
TCP467	4	●	64		cPCI	
TCP468	4		64		cPCI	
TCP469	8	●	64	●	cPCI	
TCP470	4	●	64	●	cPCI	

2 Installation

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

tdrv002bus\	Directory containing bus driver files
tdrv002port\	Directory containing serial port driver files
installer_32bit.exe	Installation tool for 32bit systems
installer_64bit.exe	Installation tool for 64bit systems
dpinst.xml	Installation XML file
tdrv002.h	Header file with IOCTL codes and structure definitions
example\tdrv002exa.c	Example application
TDRV002-SW-65-2.1.3.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 10

This section describes how to install the TDRV002-SW-65 Device Driver on a Windows 10 (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. 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 "***<Board Type> - (<n> Serial Ports Enumerator (TDRV002))***" should appear, where *<Board Type>* displays the name of the mounted board and *<n>* the number of supported serial channels.
3. Click the "**+**" in front of "***Ports (COM & LPT)***".
   The serial port devices "***<Board Type> Serial Port Device (COM<x>)***" should appear, where *<Board Type>* displays the name of the mounted board and *<x>* the assigned COM channel number.

2.3 Device and Software De-Installation

To prevent a reservation of COM-ports that will not be needed any more it is necessary to remove devices correctly. Therefor you have to decide, if the port should be removed temporarily – the board will be remounted later and the COM names should be used again – or if it should be removed permanently and the COM names can be assigned to other devices.

2.3.1 Temporary Remove and Reinstallation of Devices

Removing devices temporary is done quite simple, by a shutdown of the system and remove of the hardware. Windows will keep the configuration and naming of the devices, but the devices will not be shown.

For a reinstallation it is necessary that the hardware is mounted to the same slot again. The system will automatically start using all old configurations and names.

If the board is mounted in another slot, Windows will recognize new devices and will assign new COM names.

2.3.2 Permanent Remove of Devices

For a permanent and clean remove of serial devices it is necessary that the hardware, which shall be removed, is present in the system. The system must be started and the devices must be de-installed using the Windows Device Manager. First remove the port devices and afterwards the enumerator device. This will make sure, that the system will allow a reuse of the COM port names. After that the system should be stopped and the hardware can be removed.

If a new serial board is mounted to the system afterwards, windows will assign COM port names beginning with the first unused number, including the COM names of the de-installed ports.

2.3.3 De-installation of the Software

If the software has been installed with the installer application as described in 2.1.1, the driver should be de-installed using “Software” in the “Control Panel”. To avoid a permanent reservation of COM names, the affected devices should be removed first as described in 2.3.2.

3 Default Configuration

The default configuration of a serial 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.

3.1 Basic Port Settings

The property page allows changing the basic settings of the serial port. The basic port settings allow specifying the port setting which is used when opening the serial device.

The basic port settings contain the “Standard Serial Settings” (Baudrate, Data Bits etc.) and the “Transceiver Settings”.

The “Transceiver Settings” allow configuring the interface configuration of a port if the port supports a programmable interface. Otherwise this setting will show the supported interface and cannot be modified.

All these settings will be used to configure the port when opening. The ports may be used by application without extra configuration after opening.

Remember that applications like terminal applications usually change communication parameters after starting. The communication parameters must be configured in the terminal application.

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.

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

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

The “Advanced Port Settings” will only be used on device startup. Therefore it is necessary to restart the device after modifying any of the described settings. Restart the device using the device manager, or simply restart the system.

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 \*Device*\tdrv002_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.

On a successful execution of this function the port will be setup with the specified default port settings (See also 3.1).

```
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 Code	Description
ERROR_INVALID_PARAMETER	This function is not supported for the module type.

All other returned error codes are system error conditions.

SEE ALSO

TDRV002 Hardware User Manual

5 Known Problems

5.1 Order of Serial Ports

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