The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV002-SW-65

Windows 2000/XP Device Driver

Multiple Channel Serial Interface

Version 1.2.x

User Manual

Version 1.2.2

June 2008

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7
25469 Halstenbek, Germany
www.tews.com

Phone: +49 (0) 4101 4058 0
Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com

TEWS TECHNOLOGIES LLC
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com

Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TDRV002-SW-65

Multiple Channel Serial Interface

Windows 2000/XP Device Driver

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

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.

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

Table of Contents

1 Introduction

The TDRV002-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TDRV002 product family on an Intel or Intel-compatible x86 Windows 2000, Windows XP 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 fully compatible to the standard Windows WDM serial device driver (*serial.sys*).

Supported Modules:

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
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
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

2 Installation

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

tdrv002port.sys	Windows 2000/XP driver binary (TEWS serial channel driver)
tdrv002bus.sys	Windows 2000/XP driver binary (TEWS serial channel bus driver)
tdrv002port.inf	Windows 2000/XP installation script (channel driver)
tdrv002bus.inf	Windows 2000/XP installation script (bus driver)
tdrv002.h	Application include file
example\tdrv002exa.c	Example Code
TDRV002-SW-65-1.2.2.pdf	This document
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Software Installation

2.1.1 Windows 2000 / XP

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

After installing the hardware module(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. 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. Now a window "***Digital Signature Not Found***" will appear. To install the driver continue with installation and Click "***Yes***" button to continue.

6. 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/XP setup will show a "***New hardware found***" dialog box again.

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

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

9. Insert the TDRV002-SW-65 driver disk; select the matching drive of the distribution media in the dialog box.
   Click "***Next***" button to continue.

10. Now the driver wizard should find a suitable device driver on the distribution media.
    Click "***Next***" button to continue.

11. 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 creates devices (TDRV002_1, TDRV002_2 ...) for all recognized modules supported by the TDRV002-SW-65 device driver.

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 "***Properties***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
3. Click the "**+**" in front of "***Multi-port serial adapters***".
   The driver "***TDRV002 (TDRV002 Family Channel Enumerator)***" should appear.
4. Click the "**+**" in front of "***Ports (COM & LPT)***".
   The serial devices "***TDRV002 (UART Family) (COMxx)***" should appear.

3 FIFO Configuration

After Installation of the TDRV002-SW-65 Device Driver the trigger level for transmit and receive FIFO are set to default values. Default values are:

Receive FIFO	Transmit FIFO
56	8

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

To change the transmit trigger level the following value must be modified:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tdrv002pPort\TxFIFO

To change the receiver trigger level the following value must be modified:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\drv002pPort\RxFIFO

The trigger levels must be set to valid values between 1 and 64.

To make the changes current the system must be restarted.

4 TDRV002 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 Registry.

HKEY_LOCAL_MACHINE\Hardware\DEVICEMAP\SERIALCOMM

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

LPCTSTR 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.*

DWORD 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)

DWORD dwShareMode

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

LPSECURITY_ATTRIBUTES lpSecurityAttributes

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

DWORD 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**.

DWORD dwFlagsAndAttributes

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

HANDLE 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

HANDLE *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 write 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("\nOutput 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