The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV003-SW-65

Windows Device Driver

16(8) Digital I/O

Version 2.0.x

User Manual

Issue 2.0.0

June 2011

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

TDRV003-SW-65

Windows Device Driver

16(8) Digital I/O

Supported Modules:
TPMC670
TPMC671

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.

©2005-2011 by TEWS TECHNOLOGIES GmbH

Issue	Description	Date
1.0.0	First Issue	October 12, 2005
1.0.1	Example CloseHandle() corrected, New Address TEWS LLC, file list changed	May 19, 2008
1.0.2	Files moved to subdirectory	June 23, 2008
2.0.0	Windows7 support and API functions added	March 21, 2011

Table of Contents

1 Introduction

The TDRV003-SW-65 Windows device driver is a kernel mode driver which allows the operation of 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 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 TDRV003 device driver supports the following features:

- write new output value
- write new output value with mask
- set/clear individual output lines
- read state of input lines
- wait for interrupt events (rising/falling edge) on each input line
- start and stop the output watchdog
- acknowledge watchdog errors
- configure & start and stop input debouncing

The TDRV003-SW-65 supports the modules listed below:

TPMC670	16(8) Digital Input (24V) 16(8) Digital Output (24V, 0.5A) (50 pin connector)	PMC
TPMC671	16 Digital Input (24V) 16 Digital Output (24V, 0.5A) (64 pin connector)	PMC

In this document all supported modules and devices will be called TDRV003. Specials for certain devices will be advised.

To get more information about the features and use of TDRV003 devices it is recommended to read the manuals listed below.

TPMC670/TPMC671 User manual
TPMC670/TPMC671 Engineering Manual

2 Installation

Following files are located in directory TDRV003-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)
tdrv003.inf	Windows installation script
tdrv003.h	Header file with IOCTL codes and structure definitions
example\tdrv003exa.c	Example application
api\tdrv003api.c	Application Programming Interface source
api\tdrv003api.h	Application Programming Interface header
TDRV003-SW-65-2.0.0.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV003 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV003 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. Insert the TDRV003 driver media; 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 media.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Now copy all needed files (tdrv003.h and API files) to the desired target directories.

After successful installation the TDRV003 device driver will start immediately and creates devices (TDRV003_1, TDRV003_2 ...) for all recognized TDRV003 modules.

2.1.2 Windows 7

This section describes how to install the TDRV003-SW-65 Device Driver on a Windows 7 (32bit or 64bit) operating system.

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

Copy needed files (tdrv003.h and API files) to desired target directory.

After successful installation a device is created for each module found (TDRV003_1, TDRV003_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 driver "***TEWS TECHNOLOGIES – TDRV003 Digital I/O (TPMC670)***" should appear for each installed device.

3 Driver Configuration

3.1 Event Queue Configuration

After Installation of the TDRV003 Device Driver the number concurrent event controlled read request is limited to 10.

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 queue entries the following value must be modified.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tdrv003\Parameters\NumReqEntries

Valid values are in range between 1 and 100.

4 API Documentation

4.1 General Functions

4.1.1 tdrv003Open

NAME

tdrv003Open – Opens a Device

SYNOPSIS

```
TDRV003_HANDLE tdrv003Open
(
      char          *DeviceName
);
```

DESCRIPTION

Before I/O can be performed to a device, a file descriptor must be opened by a call to this function.

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tdrv003Open("\\\\.\\TDRV003_1" );
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. To get extended error information, call ***GetLastError***.

ERROR CODES

All error codes are standard error codes set by the I/O system.

4.1.2 tdrv003Close

NAME

tdrv003Close – Closes a Device

SYNOPSIS

```
TDRV003_STATUS tdrv003Close
(
      TDRV003_HANDLE              hdl
);
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE hdl;
TDRV003_STATUS result;

/*
** close file descriptor to device
*/
result = tdrv003Close( hdl );

if (result != TDRV003_OK)
{
     /* handle close error */
}
```

RETURNS

On success TDRV003_OK, or an appropriate error code.

ERROR CODES

All error codes are standard error codes set by the I/O system.

4.2 Device Access Functions

4.2.1 tdrv003InputRead

NAME

tdrv003InputRead – read state of input lines

SYNOPSIS

```
TDRV003_STATUS tdrv003InputRead
(
    TDRV003_HANDLE          hdl,
    unsigned short          *pInputBuf
);
```

DESCRIPTION

This function reads the current state of the input lines.

PARAMETERS

hdl

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

pInputBuf

This argument points to a buffer where the value will be returned.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;
unsigned short   data;


result = tdrv003InputRead(hdl, &data);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

4.2.2 tdrv003OutputWrite

NAME

tdrv003OutputWrite – write a new value to the output port

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputWrite
(
      TDRV003_HANDLE      hdl,
      unsigned short      OutputValue
);
```

DESCRIPTION

This function writes a new value to the output port.

PARAMETERS

hdl

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

OutputValue

This argument specifies the new output value. Bit 0 specifies the new state of output line 1; bit 1 specifies the new state for output line 2 and so on.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;

// set OUTPUT1 and OUTPUT16 and clear all other
result = tdrv003OutputWrite(hdl, 0x8001);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

4.2.3 tdrv003OutputRead

NAME

tdrv003OutputRead – read current state of output lines

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputRead
(
    TDRV003_HANDLE      hdl,
    unsigned short      *pOutputBuf
);
```

DESCRIPTION

This function reads the current state of the output lines.

PARAMETERS

hdl

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

pOutputBuf

This argument points to a buffer where the value will be returned.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;
unsigned short   data;


result = tdrv003OutputRead(hdl, &data);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

4.2.4 tdrv003OutputWriteMask

NAME

tdrv003OutputWriteMask – writes a masked value to the output port

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputWriteMask
(
    TDRV003_HANDLE      hdl,
    unsigned short      OutputValue,
    unsigned short      mask
);
```

DESCRIPTION

This control function writes a masked value to the output port. Only those bits in value for which the corresponding bit position in the mask is set to 1 will be changed in the output port.

PARAMETERS

hdl

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

OutputValue

This argument specifies the masked value for the output port. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on.

mask

This argument specifies the mask for relevant bits. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on.
Only those bits in *OutputValue* for which the corresponding bit position in this mask is set to 1 will be changed in the output port.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE  hdl;
TDRV003_STATUS  result;

// clear OUTPUT1 and set OUTPUT16 and leave all other lines unchanged
result = tdrv003OutputWriteMask(hdl, 0x8000, 0x8001);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

4.2.5 tdrv003OutputSetBits

NAME

tdrv003OutputSetBits – set specific output lines

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputSetBits
(
      TDRV003_HANDLE          hdl,
      unsigned short          OutputBits
);
```

DESCRIPTION

This function sets specific bits in the output port to active state (1).

PARAMETERS

hdl

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

OutputBits

This argument specifies a mask of relevant bits to set. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on. Bits which are set (1) in this mask will be set in the corresponding bits in the output port.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE  hdl;
TDRV003_STATUS  result;

// set OUTPUT1 and OUTPUT16 to active state
result = tdrv003OutputSetBits(hdl, (1<<15) | (1<<0));

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

4.2.6 tdrv003OutputClearBits

NAME

tdrv003OutputClearBits – clear specific output lines

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputClearBits
(
      TDRV003_HANDLE          hdl,
      unsigned short          OutputBits
);
```

DESCRIPTION

This function clears specific bits in the output port to inactive state (0).

PARAMETERS

hdl

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

OutputBits

This argument specifies a mask of relevant bits to clear. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on. Bits which are set (1) in this mask will be cleared in the corresponding bits in the output port.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE  hdl;
TDRV003_STATUS  result;

// clear OUTPUT1 and OUTPUT16
result = tdrv003OutputSetBits(hdl, (1<<15) | (1<<0));

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

4.2.7 tdrv003EventWait

NAME

tdrv003EventWait – wait for a specific input event

SYNOPSIS

```
TDRV003_STATUS tdrv003EventWait
(
    TDRV003_HANDLE      hdl,
    unsigned short      mode,
    unsigned short      mask,
    long                timeout
);
```

DESCRIPTION

This function waits for an event (rising or falling edge or both) at the specified input lines. The function is blocked until at least one of the specified events or a timeout occurs.

PARAMETERS

hdl

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

mode

This argument specifies the kind of event to wait for. The following event types are defined in tdrv003.h:

Event Type	Description
TDRV003_HIGH_TR	Wait for a low to high transition on a specified input line.
TDRV003_LOW_TR	Wait for a high to low transition on a specified input line.
TDRV003_ANY_TR	Wait for any transition on a specified input line.

mask

This argument specifies a bit mask of input lines to wait for the specified edge event. Bit 0 corresponds to the first input line; bit 1 corresponds to the second input line and so on.
Only one bit shall be set in the mask, otherwise the occurred event cannot be determined exactly. If more than one bit is set in the mask the function is completed the moment a relevant transition at least at one specified bit position occurs.

timeout

Specifies the amount of time (in seconds) the caller is willing to wait for the specified event to occur. A value of 0 means wait indefinitely.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE  hdl;
TDRV003_STATUS  result;

// wait at least 5 s for any edge at INPUT16
result = tdrv003EventWait (hdl, TDRV003_ANY_TR, 1<<15, 5);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_BUSY	No more free entries in the drivers queue to handle concurrent event-controlled read requests. Increase the queue size (see also 3.1).
TDRV003_ERR_TIMEOUT	The requested event does not occur within the specified time (timeout).

4.2.8 tdrv003DebouncerEnable

NAME

tdrv003DebouncerEnable – configure and enable debouncer circuit

SYNOPSIS

```
TDRV003_STATUS tdrv003DebouncerEnable
(
    TDRV003_HANDLE          hdl,
    unsigned short          DebounceTimer
);
```

DESCRIPTION

This function configures and enables the input debouncer circuit.

PARAMETERS

hdl

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

DebounceTimer

This argument specifies the debouncer time. The debouncer time is specified in approx. 7µs steps. Please refer to the corresponding Hardware User Manual for a calculation formula and a table of values.

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE  hdl;
TDRV003_STATUS  result;

// Enable the debouncer with a debounce time of 1ms
result = tdrv003DebouncerEnable(hdl, 147);

if (result != TDRV003_OK)
{
    /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

4.2.9 tdrv003DebouncerDisable

NAME

tdrv003DebouncerDisable – disable debouncer circuit

SYNOPSIS

```
TDRV003_STATUS tdrv003DebouncerDisable
(
      TDRV003_HANDLE              hdl
);
```

DESCRIPTION

This function disables the input debouncer circuit.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Disable the debouncer circuit
result = tdrv003DebouncerDisable(hdl);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

4.2.10 tdrv003WatchdogEnable

NAME

tdrv003WatchdogEnable – enable output watchdog

SYNOPSIS

```
TDRV003_STATUS tdrv003WatchdogEnable
(
      TDRV003_HANDLE              hdl
);
```

DESCRIPTION

This function enables the watchdog timer for the output lines. The watchdog function is activated after the next write operation to the device. Please remember that if the watchdog is enabled and no write access occurs within 120 ms, all outputs go into the inactive (0) state. To unlock the output register and leave the inactive state the function *tdrv003WatchdogReset* must be executed.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Enable output watchdog
result = tdrv003WatchdogEnable(hdl);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

4.2.11 tdrv003WatchdogDisable

NAME

tdrv003WatchdogDisable – disable output watchdog

SYNOPSIS

```
TDRV003_STATUS tdrv003WatchdogDisable
(
      TDRV003_HANDLE              hdl
);
```

DESCRIPTION

This function disables the watchdog timer for the output lines.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Disable output watchdog
result = tdrv003WatchdogDisable(hdl);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

4.2.12 tdrv003WatchdogReset

NAME

tdrv003WatchdogReset – reset output watchdog error

SYNOPSIS

```
TDRV003_STATUS tdrv003WatchdogReset
(
      TDRV003_HANDLE              hdl
);
```

DESCRIPTION

This device function resets an output watchdog error. This function must be called after a device function returns the error code *TDRV003_ERR_IO*.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tdrv003api.h"

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Reset watchdog error
result = tdrv003WatchdogReset (hdl);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

5 Device Driver Programming

The TDRV003-SW-65 Windows device driver is a kernel mode device driver.

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.

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.)

5.1 TDRV003 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TDRV003 device driver. Only the required parameters are described in detail.

5.1.1 Opening a TDRV003 Device

Before you can perform any I/O the *TDRV003* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TDRV003* 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 TDRV003 to open. The *lpFileName* string should be of the form **\\.\TDRV003_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TDRV003_1, the second \\.\TDRV003_2 and so on.

dwDesiredAccess

This parameter specifies the type of access to the TDRV003.
For the TDRV003 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 TDRV003 devices.

dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TDRV003 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 TDRV003 devices.

RETURN VALUE

If the function succeeds, the return value is an open handle to the specified TDRV003 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

EXAMPLE

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TDRV003_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                    // no security attrs
     OPEN_EXISTING,           // TDRV003 device always open existing
     0,
     NULL
);

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

SEE ALSO

CloseHandle(), Win32 documentation CreateFile()

5.1.2 Closing a TDRV003 Device

The **CloseHandle** function closes an open TDRV003 handle.

BOOL CloseHandle
(
HANDLE *hDevice;*
);

PARAMETERS

hDevice

Identifies an open TDRV003 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 ()

5.1.3 TDRV003 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 TDRV003 that is to perform the operation.

dwIoControlCode

This argument specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in *tdrv003.h*:

Value	Meaning
IOCTL_TDRV003_READ	Read input port immediately
IOCTL_TDRV003_WRITE	Write output port
IOCTL_TDRV003_OUTPUTGET	Read current output value
IOCTL_TDRV003_WRITEMASK	Write output port with mask
IOCTL_TDRV003_OUTPUTSETBITS	Set specific output lines
IOCTL_TDRV003_OUTPUTCLEARBITS	Clear specific output lines
IOCTL_TDRV003_WAIT_EVENT	Wait for a specified event
IOCTL_TDRV003_DEBENABLE	Enable input debounce function
IOCTL_TDRV003_DEBDISABLE	Disable input debounce function
IOCTL_TDRV003_WDENABLE	Enable output watchdog
IOCTL_TDRV003_WDDISABLE	Disable output watchdog
IOCTL_TDRV003_WDRESET	Reset output watchdog

See below for more detailed information on each control code.

To use these TDRV003 specific control codes the header file tdrv003.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

This argument 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

This argument is a pointer to an *Overlapped* structure. This value must be set to NULL (no overlapped I/O).

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()

5.1.3.1 IOCTL_TDRV003_READ

This control function reads the current state of the input port.

The content is returned in an unsigned short buffer pointed by *lpOutBuffer*. The buffer must be always an unsigned short type independent of the TDRV003 variant. The argument *nOutBufferSize* specifies the size (size of USHORT) of the buffer.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant.

EXAMPLE

```
#include "tdrv003.h"

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

success = DeviceIoControl(
     hDevice,
     IOCTL_TDRV003_READ,
     NULL,
     0,
     &PortData,
     sizeof(PortData),
     &NumBytes,
     NULL
);

if( success )
{
     printf ("Read input port successful (input port = 0x%x)\n",
          PortData);
}
else
{
     ErrorHandler ("Device I/O control error" ); // process error
}
```

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the read buffer is too small.

All other returned error codes are system error conditions.

5.1.3.2 IOCTL_TDRV003_WRITE

This control function writes a new value to the output port

The new port value is passed in an unsigned short buffer, pointed by *lpInBuffer,* to the driver. The buffer must be always an unsigned short type independent of the TDRV003 variant. The argument *nInBufferSize* specifies the size (size of USHORT) of the write buffer.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant

EXAMPLE

```
#include "tdrv003.h"


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


PortData = 0x1234;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_WRITE,
     &PortData,
     sizeof(PortData),
     NULL,
     0,
     &NumBytes,
     NULL
);


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

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the write buffer is too small.
ERROR_IO_DEVICE	The output register is locked by a watchdog failure. Execute the control function IOCTL_TDRV003_WDRESET to reset the watchdog error.

All other returned error codes are system error conditions.

SEE ALSO

Win32 documentation DeviceIoControl(), TDRV003 Hardware User Manual

5.1.3.3 IOCTL_TDRV003_OUTPUTGET

This control function reads the current state of the output port.

The content is returned in an unsigned short buffer pointed by *lpOutBuffer*. The buffer must be always an unsigned short type independent of the TDRV003 variant. The argument *nOutBufferSize* specifies the size (size of USHORT) of the buffer.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant.

EXAMPLE

```
#include "tdrv003.h"

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

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_OUTPUTGET,
     NULL,
     0,
     &PortData,
     sizeof(PortData),
     &NumBytes,
     NULL
);

if( success )
{
     printf ("Read output port successful (0x%x)\n", PortData);
}
else
{
     ErrorHandler ("Device I/O control error" ); // process error
}
```

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the read buffer is too small.
ERROR_IO_DEVICE	The output register is locked by a watchdog failure. Execute the control function IOCTL_TDRV003_WDRESET to reset the watchdog error.

All other returned error codes are system error conditions.

5.1.3.4 IOCTL_TDRV003_WRITEMASK

This control function writes a masked value to the output port. Only those bits in value for which the corresponding bit position in the mask is set to 1 will be changed in the output port.

The parameter *lpInBuffer* must pass a pointer to the write buffer *(TDRV003_WRITEBUF*) to the device driver. The argument *nInBufferSize* specifies the size of this buffer.

typedef struct {
    unsigned short value;
    unsigned short mask;
} TDRV003_WRITEBUF, *PTDRV003_ WRITEBUF;

value

This parameter specifies the new output value for the output port. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on.

mask

This parameter specifies the mask for relevant bits. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on.
Only those bits in value for which the corresponding bit position in this mask is set to 1 will be changed in the output port.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant

EXAMPLE

```
#include "tdrv003.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
TDRV003_WRITEBUF WriteBuf;

// set OUTPUT1 to 0 and OUTPUT16 to 1
WriteBuf.mask = 0x8001;
WriteBuf.value = 0x8000;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_WRITEMASK,
     &WriteBuf,
     sizeof(WriteBuf),
     NULL,
     0,
     &NumBytes,
     NULL
);
```

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

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the write buffer is too small.
ERROR_IO_DEVICE	The output register is locked by a watchdog failure. Execute the control function IOCTL_TDRV003_WDRESET to reset the watchdog error.

All other returned error codes are system error conditions.

SEE ALSO

Win32 documentation DeviceIoControl(), TDRV003 Hardware User Manual

5.1.3.5 IOCTL_TDRV003_OUTPUTSETBITS

This control function sets specific bits in the output port to active state (1).

A mask of relevant bits to set is passed in an unsigned short buffer, pointed by *lpInBuffer,* to the driver. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on. Bits which are set (1) in this mask will be set in the corresponding bits in the output port.

The argument *nInBufferSize* specifies the size (size of USHORT) of this variable.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant

EXAMPLE

```
#include "tdrv003.h"


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


// set OUTPUT1, OUTPUT2 and OUTPUT16 active (1)
OutputBits = (1<<15) | (1<<1) | (1<<0);

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_OUTPUTSETBITS,
     &OutputBits,
     sizeof(OutputBits),
     NULL,
     0,
     &NumBytes,
     NULL
);

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

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the write buffer is too small.
ERROR_IO_DEVICE	The output register is locked by a watchdog failure. Execute the control function IOCTL_TDRV003_WDRESET to reset the watchdog error.

All other returned error codes are system error conditions.

SEE ALSO

Win32 documentation DeviceIoControl(), TDRV003 Hardware User Manual

5.1.3.6 IOCTL_TDRV003_OUTPUTCLEARBITS

This control function clears specific bits in the output port to inactive state (0).

A mask of relevant bits to clear is passed in an unsigned short buffer, pointed by *lpInBuffer,* to the driver. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on. Bits which are set (1) in this mask will be cleared in the corresponding bits in the output port.

The argument *nInBufferSize* specifies the size (size of USHORT) of this variable.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant

EXAMPLE

```
#include "tdrv003.h"


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


// set OUTPUT1, OUTPUT2 and OUTPUT16 inactive (0)
OutputBits = (1<<15) | (1<<1) | (1<<0);

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_OUTPUTCLEARBITS,
     &OutputBits,
     sizeof(OutputBits),
     NULL,
     0,
     &NumBytes,
     NULL
);

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

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the write buffer is too small.
ERROR_IO_DEVICE	The output register is locked by a watchdog failure. Execute the control function IOCTL_TDRV003_WDRESET to reset the watchdog error.

All other returned error codes are system error conditions.

SEE ALSO

Win32 documentation DeviceIoControl(), TDRV003 Hardware User Manual

5.1.3.7 IOCTL_TDRV003_WAIT_EVENT

This control function waits for an event (rising or falling edge or both) at the specified input lines. The function is blocked until at least one of the specified events or a timeout occurs.

The parameter *lpInBuffer* must pass a pointer to the buffer *(TDRV003_WAIT_BUFFER*) to the device driver. The argument *nInBufferSize* specifies the size of this buffer.

```
typedef struct {
    unsigned short      mode;
    unsigned short      mask;
    long                timeout;
} TDRV003_WAIT_BUFFER, *PTDRV003_WAIT_BUFFER;
```

mode

This parameter specifies the kind of event to wait for. The following event types are defined in tdrv003.h:

Event Type	Description
TDRV003_HIGH_TR	Wait for a low to high transition on a specified input line.
TDRV003_LOW_TR	Wait for a high to low transition on a specified input line.
TDRV003_ANY_TR	Wait for any transition on a specified input line.

mask

This parameter specifies a bit mask of input lines to wait for the specified edge event. Bit 0 corresponds to the first input line; bit 1 corresponds to the second input line and so on.
Only one bit shall be set in the mask, otherwise the occurred event cannot be determined exactly. If more than one bit is set in the mask the function is completed at the moment a relevant transition at least at one specified bit position occurs.

timeout

Specifies the amount of time (in seconds) the caller is willing to wait for the specified event to occur. A value of 0 means wait indefinitely.

EXAMPLE

```
#include "tdrv003.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
TDRV003_WAIT_BUFFER WaitBuf;

/*
**   Wait for a high-transition at bit 7 (INPUT 8)
*/
WaitBuf.mode       = TDRV003_HIGH_TR;
WaitBuf.mask       = 1<<7;        // high-transition at bit 7
WaitBuf.timeout    = 10;          // seconds
```

```
success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_WAIT_EVENT,
     &WaitBuf,
     sizeof(TDRV003_WAIT_BUFFER),
     NULL,
     0,
     &NumBytes,
     NULL
);
if( success )
{
     printf("Success\n");
}
else
{
     ErrorHandler ("Device I/O control error"); // process error
}
```

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the wait buffer is too small or if the parameter mode contains an invalid value.
ERROR_NO_SYSTEM_RESOURCES	No more free entries in the drivers queue to handle concurrent event-controlled read requests. Increase the queue size (see also 3.1).
ERROR_SEM_TIMEOUT	The requested event does not occur within the specified time (timeout).

All other returned error codes are system error conditions.

5.1.3.8 IOCTL_TDRV003_DEBENABLE

This control function configures and enables the input debouncer function.

The new debounce timer value is passed by an unsigned short variable, pointed by *lpInBuffer*, to the driver. The argument *nInBufferSize* specifies the size (size of USHORT) of this variable.

See also TDRV003 Hardware User Manual – Debounce Time Register for counter calculation formulas.

EXAMPLE

```
#include "tdrv003.h"

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


/*
**  Enable the debouncer with a debounce time of 1ms
*/
DebounceTime = 147;

success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_DEBENABLE,
     &DebounceTime,
     sizeof(DebounceTime),
     NULL,
     0,
     &NumBytes,
     NULL
);

if( success )
{
     printf("Enable output watchdog successful\n");
} else
{
     ErrorHandler ("Device I/O control error"); // process error
}
```

ERROR CODES

ERROR_INVALID_PARAMETER	This error is returned if the size of the timer value buffer is too small

All other returned error codes are system error conditions.

5.1.3.9 IOCTL_TDRV003_DEBDISABLE

This control function disables the input debouncer function enabled by *IOCTL_TDRV003_DEBENABLE*.

No additional parameter is required for this function.

EXAMPLE

```
#include "tdrv003.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_DEBDISABLE,
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL
);

if( success )
{
     printf("Disable debouncer successful\n");
}
else
{
     ErrorHandler ("Device I/O control error"); // process error
}
```

ERROR CODES

No driver specific error codes

5.1.3.10 IOCTL_TDRV003_WDENABLE

This control function enables the output watchdog function after the next write operation to the device. Please remember if the watchdog is enabled and no write access occurs within 120 ms all outputs go into the inactive state. To unlock the output register and leave the inactive state the device control function *IOCTL_TDRV003_WDRESET* must be executed.

No additional parameter is required for this function.

EXAMPLE

```
#include "tdrv003.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_WDENABLE,
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL
);

if( success )
{
     printf("Enable output watchdog successful\n");
}
else
{
     ErrorHandler ("Device I/O control error"); // process error
}
```

ERROR CODES

No driver specific error codes

5.1.3.11 IOCTL_TDRV003_WDDISABLE

This device control function disables the output watchdog.

No additional parameter is required for this function.

EXAMPLE

```
#include "tdrv003.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_WDDISABLE,
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL
);

if( success )
{
     printf("Disable output watchdog successful\n");
}
else
{
     ErrorHandler ("Device I/O control error"); // process error
}
```

ERROR CODES

No driver specific error codes

5.1.3.12 IOCTL_TDRV003_WDRESET

This device control function resets an output watchdog error. This function must be called after a device control function returns the error code *ERROR_IO_DEVICE.*

No additional parameter is required for this function.

EXAMPLE

```
#include "tdrv003.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,
     IOCTL_TDRV003_WDRESET,
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL
);

if( success )
{
     printf("Reset output watchdog successful\n");
}
else
{
     ErrorHandler ("Device I/O control error"); // process error
}
```

ERROR CODES

No driver specific error codes