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 2.0.1

March 2024

TEWS Technologies GmbH
Eggerstedter Weg 14, 25421 Pinneberg, Germany
Phone: +49 (0) 4101 4058 0
e-mail: info@tews.com www.tews.com

TDRV003-SW-65

Windows Device Driver

16(8) Digital I/O

Supported Modules:
TPMC670
TPMC671

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
2.0.1	Support for Windows 10 added, Installation modified, Description of Ioctl functions removed, New address TEWS Technologies GmbH	March 7, 2024

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 systems.

The TDRV003-SW-65 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

2 Installation

Following files are located in directory TDRV003-SW-65 on the distribution media:

driver\	Directory containing driver files
api\tdrv003api.c	Application Programming Interface source
api\tdrv003api.h	Application Programming Interface header
tdrv003.h	Header file with IOCTL codes and structure definitions
example\tdrv003exa.c	Example application
installer_32bit.exe	Installation tool for 32bit systems
installer_64bit.exe	Installation tool for 64bit systems
dpinst.xml	Installation XML file
TDRV003-SW-65-2.0.1.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 TDRV003-SW-65 Device Driver on a Windows 10 (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, API files) to desired target directory.

After successful installation a device is created for each module found (TDRV003_1, TDRV003_2 ...).

2.2 Confirming Windows Driver Installation

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

1. Open the Windows Device Manager:
   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 (e.g. "***TPMC670 Digital I/O (TDRV003)***") 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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

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

Error Code	Description
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.

Other returned error codes are system error conditions.

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

Error Code	Description
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.

Other returned error codes are system error conditions.

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

Error Code	Description
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.

Other returned error codes are system error conditions.

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

Error Code	Description
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.

Other returned error codes are system error conditions.

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

Error Code	Description
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.

Other returned error codes are system error conditions.

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

Error Code	Description
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).

Other returned error codes are system error conditions.

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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

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

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.