The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV003-SW-42

VxWorks Device Driver

16 (8) Digital Inputs and 16 (8) Digital Outputs

Version 5.1.x

User Manual

Issue 5.1.0

March 2022

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

TDRV003-SW-42

VxWorks Device Driver

16 (8) Digital Inputs and 16 (8) Digital Outputs

Supported Modules:
TPMC670
TPMC671

Issue	Description	Date
1.0.0	First Issue	October 14, 2005
1.0.1	New Address TEWS LLC, general revision	October 21, 2008
2.0.0	API documentation added	September 6, 2010
2.0.1	General Revision	January 25, 2011
3.0.0	VxWorks 6.9 support, 64-bit support	February 07, 2012
4.0.0	API modified, Basic I/O functions removed	July 04, 2013
5.0.0	VxWorks 7 support, new installation guide	November 13, 2017
5.1.0	ioctl for RTP-Support modified	March 30, 2022

Table of Contents

1 Introduction

The TDRV003-SW-42 VxWorks device driver software allows the operation of the modules supported by TDRV003 conforming to the VxWorks I/O system specification.

The TDRV003-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new VxBus-enabled (GEN1 or GEN2) driver model. The VxBus-enabled driver is recommended for new developments with later VxWorks 6.x and 7.x releases and mandatory for VxWorks SMP systems.

Both drivers, legacy and VxBus, share the same application programming interface (API) and invoke a mutual exclusion and binary semaphore mechanism to prevent simultaneous requests by multiple tasks from interfering with each other.

The TDRV003-SW-42 device driver supports the following features:

- read the actual input value
- write a new output value
- wait for selectable input events (match, high-, low-, any transition on the input line(s))
- enable and disable the output watchdog
- acknowledge watchdog errors
- configure, enable and disable input debouncing

The TDRV003-SW-42 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.

TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide
TPMC670/TPMC671 User Manual

2 API Documentation

2.1 General Functions

2.1.1 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. The first TDRV003 device is named “/tdrv003/0”, the second device is named “/tdrv003/1” and so on.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tdrv003Open(“/tdrv003/0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device descriptor pointer, or NULL if the function fails. An error code will be stored in *errno*.

ERROR CODES

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

2.1.2 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.

2.2 Device Access Functions

2.2.1 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.

2.2.2 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.

2.2.3 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.

2.2.4 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.

2.2.5 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.

2.2.6 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 = tdrv003OutputClearBits(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.

2.2.7 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 tdrv003api.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 milliseconds) 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, 5000);

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 chapter 3.3).
TDRV003_ERR_TIMEOUT	The requested event does not occur within the specified time (timeout).

2.2.8 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.

2.2.9 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.

2.2.10 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.

2.2.11 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.

2.2.12 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.

3 Appendix

3.1 Enable RTP-Support

Using TDRV003 devices tunneled from Real Time Processes (RTPs) is implemented. For this the “TEWS TDRV003 IOCTL command validation” must be enabled in system configuration.

The API source file “tdrv003api.c” must be added to the RTP-Project directory and built together with the RTP-application.

The definition of TVXB_RTP_CONTEXT must be added to the project, which is used to eliminate kernel headers, values and functions from the used driver files.

Find more detailed information in “TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide”.

All legacy functions, functions for version compatibility and debugging functions are not usable from RTPs.

3.2 Debugging and Diagnostic

The TDRV003 device driver provides a function, and debug statements to display versatile information of the driver installation and status on the debugging console.

If the VxBus driver is used, the TDRV003 show routine is included in the driver by default and can be called from the VxWorks shell. If this function is not needed or program space is rare the function can be removed from the code by un-defining the macro INCLUDE_TDRV003_SHOW in tdrv003drv.c

The tdrv003Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TDRV003 modules and device statistics.

If TDRV003 modules were probed but no devices were created it may be helpful to enable debugging code inside the driver code by defining the macro TDRV003_DEBUG in tdrv003drv.c.

In contrast to VxBus TDRV003 devices, legacy TDRV003 devices must be created "manually". This will be done with the first call to the tdrv003Open API function.

```
-> tdrv003Show
Probed Modules:
    [0] TPMC670: Bus=4, Dev=0, DevId=0x9050, VenId=0x10b5, Init=OK, vxDev=0xffff800000042560
    [1] TPMC671: Bus=4, Dev=1, DevId=0x029f, VenId=0x1498, Init=OK, vxDev=0xffff800000043550

Associated Devices:
    [0] TPMC670: /tdrv003/0
    [1] TPMC671: /tdrv003/1

Device Statistics:
    /tdrv003/0:
        open count           = 0
        interrupt count      = 0
        Current Input Value  = 0x0000
        Current Output Value = 0x0000
        Active pending Jobs  = 0
        Debouncer disabled
        Watchdog disabled
    /tdrv003/1:
        open count           = 0
        interrupt count      = 0
        Current Input Value  = 0x0000
        Current Output Value = 0x0000
        Active pending Jobs  = 0
        Debouncer disabled
        Watchdog disabled
value = 26 = 0x1a
```

3.3 Driver Configurations

The value *TDRV003_NUMJOBS* (defined in *tdrv003def.h*) specifies the maximum number of jobs that can be handled at the same time for a device. System resources will be allocated while driver start to prevent extra time during normal application operation. This value may be changed to increase the maximum number of jobs or decreased to save system resources.

The driver must be built again after changing *TDRV003_NUMJOBS*.