The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV006-SW-42

VxWorks Device Driver

64 Digital Inputs/Outputs (Bit I/O)

Version 4.2.x

User Manual

Issue 4.2.0

March 2021

powerBridge Computer

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

www.powerbridge.de
info@powerbridge.de

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

TDRV006-SW-42

VxWorks Device Driver

64 Digital Inputs/Outputs (Bit I/O)

Supported Modules:
TPMC681

Issue	Description	Date
1.0.0	First Issue	July 7, 2006
1.0.1	New Address TEWS LLC, ChangeLog.txt added in file list	December 3, 2006
2.0.0	New ioctl() function: FIO_TDRV006_WRITE_MASKED, description for API and VxBus-support added, Address TEWS LLC removed	January 26, 2010
2.0.1	Legacy vs. VxBus Driver modified	March 26, 2010
2.1.0	New function tdrv006Init()	November 8, 2011
2.1.1	Formation of Document corrected	November 9, 2011
3.0.0	API changed (header file, return value, handle) Legacy I/O function description removed	February 20, 2012
4.0.0	VxWorks 7 support, new installation guide	December 18, 2017
4.1.0	VxWorks 7 (R2) support	May 13, 2019
4.2.0	ioctl for RTP-Support modified	March 29, 2021

Table of Contents

1 Introduction

1.1 Device Driver

The TDRV006-SW-42 VxWorks device driver software allows the operation of the TPMC681 PMC conforming to the VxWorks I/O system specification.

The TDRV006-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new VxBus-enabled (GEN1 + 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 TDRV006-SW-42 device driver supports the following features:

- configure direction of I/O lines
- set output value of output lines
- read value of I/O lines
- wait for input line events

The TDRV006-SW-42 supports the modules listed below:

TPMC681	64 bit digital I/O	PMC

In this document all supported modules and devices will be called TDRV006. Specials for certain devices will be advised.

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

TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide
TPMC681 User Manual

2 API Documentation

2.1 General Functions

2.1.1 tdrv006Open

NAME

tdrv006Open() – open a device.

SYNOPSIS

```
TDRV006_HANDLE tdrv006Open
(
    char        *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a file descriptor must be opened by a call to this function. If the legacy TDRV006 driver is used, this function will also install the legacy driver and create devices with the first call. The VxBus TDRV006 driver will be installed automatically by the VxBus system.

The tdrv006Open function can be called multiple times (e.g. in different tasks).

PARAMETER

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TDRV006 device is named “/tdrv006/0”, the second device is named “/tdrv006/1” and so on.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tdrv006Open("/tdrv006/0");
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

The error codes are stored in *errno.*

The error code is a standard error code set by the I/O system.

2.1.2 tdrv006Close

NAME

tdrv006Close() – close a device.

SYNOPSIS

```
TDRV006_STATUS tdrv006Close
(
      TDRV006_HANDLE      hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETER

hdl

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

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE      hdl;
TDRV006_STATUS      result;

/*
** close file descriptor to device
*/
result = tdrv006Close(hdl);
if (result != TDRV006_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2 Device Access Functions

2.2.1 tdrv006Read

NAME

tdrv006Read – read current input value of the I/O lines

SYNOPSIS

```
TDRV006_STATUS tdrv006Read
(
        TDRV006_HANDLE      hdl,
        unsigned int        *in31_0,
        unsigned int        *in63_32
)
```

DESCRIPTION

This function reads the current input value of the I/O lines.

PARAMETER

hdl

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

in31_0

This argument points to a buffer where the current value of I/O lines 0 up to 31 will be returned. Bit 0 returns the value of I/O line 0, bit 1 the value of I/O line 1, and so on.

in63_32

This argument points to a buffer where the current value of I/O lines 32 up to 63 will be returned. Bit 0 returns the value of I/O line 32, bit 1 the value of I/O line 33, and so on.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;
UINT32             in_low;
UINT32             in_high;

/*
** read current state of I/O lines
*/
result = tdrv006Read(hdl, &in_low, &in_high);
if (result != TDRV006_OK)
{
     /* handle error */
}
else
{
     printf("INPUT: 0x%08X%08X\n", in_high, in_low);
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVAL	A NULL pointer is referenced for an input value
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2.2 tdrv006Write

NAME

tdrv006Write – set output value

SYNOPSIS

```
TDRV006_STATUS tdrv006Write
(
        TDRV006_HANDLE      hdl,
        unsigned int        out31_0,
        unsigned int        out63_32
)
```

DESCRIPTION

This function sets the output value.

The specified value will only appear on the I/O lines which are configured for output.

PARAMETER

hdl

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

out31_0

This argument specifies the output value for I/O lines 0 up to 31. Bit 0 specifies the value of I/O line 0, bit 1 the value of I/O line 1, and so on.

out63_32

This argument specifies the output value for I/O lines 32 up to 63. Bit 0 specifies the value of I/O line 32, bit 1 the value of I/O line 33, and so on.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Set output value (set I/O lines 0-15)
*/
result = tdrv006Write(hdl, 0x0000FFFF, 0x00000000);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2.3 tdrv006WriteMasked

NAME

tdrv006WriteMasked – set output value for specified I/O lines

SYNOPSIS

```
TDRV006_STATUS tdrv006WriteMasked
(
        TDRV006_HANDLE      hdl,
        unsigned int        out31_0,
        unsigned int        out63_32,
        unsigned int        mask31_0,
        unsigned int        mask63_32
)
```

DESCRIPTION

This function sets the output value for specified I/O lines. The mask specifies which I/O bits shall be set to the specified output value and which shall keep the current value.

This specified value will only appear on the I/O lines which are configured for output.

PARAMETER

hdl

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

out31_0

This argument specifies the output value for I/O lines 0 up to 31. Bit 0 specifies the value of I/O line 0, bit 1 the value of I/O line 1, and so on.

out63_32

This argument specifies the output value for I/O lines 32 up to 63. Bit 0 specifies the value of I/O line 32, bit 1 the value of I/O line 33, and so on.

mask31_0

This argument specifies the output mask for output lines 0 up to 31. Bit 0 specifies the mask for I/O line 0, bit 1 the value for I/O line 1, and so on.
A set bit (1) means the bit shall be set to the value specified by *out31_0*.
A reset bit (0) means that the old output value will not be changed.

mask63_32

This argument specifies the output mask for output lines 32 up to 63. Bit 0 specifies the mask for I/O line 32, bit 1 the value for I/O line 33, and so on.
A set bit (1) means the bit shall be set to the value specified by *out63_32.*
A reset bit (0) means that the old output value will not be changed.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Set a part of the output value (set/reset I/O lines 0-15 and 48-63)
*/
result = tdrv006WriteMasked(hdl,
                            0x12345678, 0x87654321,
                            0x0000FFFF, 0xFFFF0000);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2.4 tdrv006SetOutputLine

NAME

tdrv006SetOutputLine – set a specified output line

SYNOPSIS

```
TDRV006_STATUS tdrv006SetOutputLine
(
        TDRV006_HANDLE      hdl,
        int                 outputLine
)
```

DESCRIPTION

This function sets a single bit of the output value.

This specified value will only appear if the corresponding I/O line is configured for output.

PARAMETER

hdl

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

outputLine

This argument specifies a data bit that shall be set. Allowed values are 0 up to 63.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Set I/O line 32
*/
result = tdrv006SetOutputLine(hdl, 32);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVAL	An invalid line number is specified
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2.5 tdrv006ClearOutputLine

NAME

tdrv006ClearOutputLine – reset a specified I/O line

SYNOPSIS

```
TDRV006_STATUS tdrv006ClearOutputLine
(
    TDRV006_HANDLE    hdl,
    int               outputLine
)
```

DESCRIPTION

This function resets a single bit of the output value.

This specified value will only appear if the corresponding I/O line is configured for output.

PARAMETER

hdl

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

outputLine

This argument specifies a data bit that shall be reset. Allowed values are 0 up to 63.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Clear I/O line 32
*/
result = tdrv006ClearOutputLine(hdl, 32);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVAL	An invalid line number is specified
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2.6 tdrv006OutputEnable

NAME

tdrv006OutputEnable – set the I/O line direction

SYNOPSIS

```
TDRV006_STATUS tdrv006OutputEnable
(
    TDRV006_HANDLE      hdl,
    unsigned int        enaout31_0,
    unsigned int        enaout63_32
)
```

DESCRIPTION

This function sets the I/O line direction. The value specifies which I/O lines shall be configured for output and which I/O lines should be used for input.

PARAMETER

hdl

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

enaout31_0

This argument specifies the direction of I/O lines 0 up to 31. Bit 0 specifies the direction of I/O line 0, bit 1 the direction of I/O line 1, and so on. A set bit (1) configures the line for output, an unset bit (0) configures input (tri-state).

enaout63_32

This argument specifies the direction of I/O lines 32 up to 63. Bit 0 specifies the direction of I/O line 32, bit 1 the direction of I/O line 33, and so on. A set bit (1) configures the line for output, an unset bit (0) configures input (tri-state).

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Enable I/O lines 0-8 for ouput
*/
result = tdrv006OutputEnable(hdl, 0x000001FF, 0x00000000);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

2.2.7 tdrv006WaitForLowToHigh

NAME

tdrv006WaitForLowToHigh – wait until a low to high transition occurs

SYNOPSIS

```
TDRV006_STATUS tdrv006WaitForLowToHigh
(
    TDRV006_HANDLE    hdl,
    int               inputLine,
    int               timeout
)
```

DESCRIPTION

This function waits until a low to high transition occurs on the specified input line or the specified timeout time has passed.

PARAMETER

hdl

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

inputLine

This argument specifies the input line which shall be observed for a low to high transition. Allowed values are 0 up to 63.

timeout

This argument specifies the time the function is willing to wait for the specified transition. If the specified time has passed the function will return with an error. The timeout is specified in milliseconds.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Wait for a low-to-high transition on input line 0
** Timeout after 1000 ms
*/
result = tdrv006WaitForLowToHigh (hdl, 0, 1000);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVAL	Invalid parameter specified
TDRV006_ERR_BUSY	Another task is already waiting for a transition of the specified input line
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid
TDRV006_ERR_TIMEOUT	Timeout occurred.

2.2.8 tdrv006WaitForHighToLow

NAME

tdrv006WaitForHighToLow – wait until a high to low transition occurs

SYNOPSIS

```
TDRV006_STATUS tdrv006WaitHighToLow
(
    TDRV006_HANDLE    hdl,
    int               inputLine,
    int               timeout
)
```

DESCRIPTION

This function waits until a high to low transition occurs on the specified input line or the specified timeout time has passed.

PARAMETER

hdl

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

inputLine

This argument specifies the input line which shall be observed for a high to low transition. Allowed values are 0 up to 63.

timeout

This argument specifies the time the function is willing to wait for the specified transition. If the specified time has passed the function will return with an error. This time is specified in milliseconds.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Wait for a high-to-low transition on input line 0
** Timeout after 1000 ms
*/
result = tdrv006WaitForHighToLow (hdl, 0, 1000);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVAL	Invalid parameter specified
TDRV006_ERR_BUSY	Another task is already waiting for a transition of the specified input line
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid
TDRV006_ERR_TIMEOUT	Timeout occurred.

2.2.9 tdrv006WaitForAnyTrans

NAME

tdrv006WaitForAnyTrans – wait until a transition occurs

SYNOPSIS

```
TDRV006_STATUS tdrv006ForAnyTrans
(
    TDRV006_HANDLE    hdl,
    int               inputLine,
    int               timeout
)
```

DESCRIPTION

This function waits until a transition (high to low or low to high) occurs on the specified input line or the specified timeout time has passed.

PARAMETER

hdl

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

inputLine

This argument specifies the input line which shall be observed for a transition. Allowed values are 0 up to 63.

timeout

This argument specifies the time the function is willing to wait for the specified transition. If the specified time has passed the function will return with an error. This time is specified in milliseconds.

EXAMPLE

```
#include "tdrv006api.h"

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Wait for a transition on input line 0
** Timeout after 1000 ms
*/
result = tdrv006WaitForAnyTrans (hdl, 0, 1000);
if (result != TDRV006_OK)
{
     /* error handling */
}
```

RETURN VALUE

On success, TDRV006_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV006_ERR_INVAL	Invalid parameter specified
TDRV006_ERR_BUSY	Another task is already waiting for a transition of the specified input line
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid
TDRV006_ERR_TIMEOUT	Timeout occurred.

3 Appendix

3.1 Enable RTP-Support

Using TDRV006 devices tunneled from Real Time Processes (RTPs) is implemented. For this the “TEWS TDRV006 IOCTL command validation” must be enabled in system configuration.

The API source file “tdrv006api.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 TDRV006 device driver provides functions and debug statements to display versatile information of the driver installation and status on the debugging console.

The TDRV006 VxBus device driver offers a special show function to display driver information. By default the TDRV006 show routine is included in the driver 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_TDRV006_SHOW in tdrv006drv.c

The tdrv006Show function displays detailed information about probed modules, assignment of devices respective device names to probed TDRV006 modules and device statistics.

If TDRV006 modules were probed but no devices were created it may be helpful to enable debugging code inside the driver code by defining the macro TDRV006_DEBUG in tdrv006drv.c. Certain debug information can be selected by assigning one or more (logical OR) TDRV_DBG_xxx values to variable tdrv006Debug.

In contrast to VxBus TDRV006 devices, legacy TDRV006 devices must be created "manually". This will be done with the first call to the tdrv006Open API function.

```
-> tdrv006Show
Probed Modules:
    [0] TDRV006: Bus=4, Dev=2, DevId=0x02a9, VenId=0x1498, Init=OK, vxDev=0xffff800000005380

Associated Devices:
    [0] TDRV006: /tdrv006/0

Device Statistics:
    /tdrv006/0:
        open count          = 0
        interrupt count     = 0
value = 0 = 0x0
```