The Embedded I/O Company

TDRV006-SW-25

Integrity Device Driver

64 Digital Inputs/Outputs

(Bit I/O)

Version 1.1.x

User Manual

Issue 1.1.0

March 2019

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

TDRV006-SW-25

Integrity Device Driver

64 Digital Inputs/Outputs

Supported Modules:
TPMC681

Issue	Description	Date
1.0.0	First Issue	August 25, 2014
1.1.0	Installation changed for Integrity 11 Examples changed into an interactive application	March 26, 2019

Table of Contents

1 Introduction

The TDRV006-SW-25 Integrity device driver software allows the operation of the supported PMC conforming to the Integrity I/O system specification. The software is designed and tested with Integrity 11.4.4.

The driver software uses mutual exclusion to prevent simultaneous requests by multiple tasks from interfering with each other.

The TDRV006-SW-25 device driver supports the following features:

- read state of the I/O lines
- write (masked) to the output register
- set/reset single output bits
- configure line direction (enable output)
- wait for a specified transition on an I/O line

The TDRV006-SW-25 supports the modules listed below:

TPMC681	64 Digital Inputs/Outputs	(PMC)

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

TPMC681 User Manual

2 Installation

The following files are located on the distribution media:

Directory path TDRV006-SW-25:

tdrv006.c	Device driver source
tdrv006def.h	Driver include file
tdrv006.h	Include file for driver and application
tdrv006api.c	Application interface, simplifies device access
tdrv006api.h	Include file for API and applications
example/tdrv006exa.c	Example application
TDRV006-SW-25-1.1.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Driver Installation

Copy the TDRV006 driver files into a desired driver or project path. The driver source file tdrv006.c must be included into the kernel project and the BSP paths must be added to the include search paths of the file. Set Options… ➔ Project → Include Directories, then double click and add the new paths:

```
$(__OS_DIR)/bsp
$(__OS_DIR)/system
$(__OS_DIR)/modules/ghs/bspsrc
$(__OS_DIR)/modules/ghs/bspsrc/support
$(__OS_DIR)/modules/ghs/bspsrc/driver/busspace
```

Afterwards the project must be rebuilt. The driver will be started automatically after booting the image and the driver will be requested if a matching device is detected in the system.

For further information building a kernel, please refer to Greenhills MULTI and INTEGRITY Documentation.

2.2 TDRV006 Applications

Copy the TDRV006 API files (tdrv006api.c, tdrv006api.h, and tdrv006.h) into a desired application path, and include tdrv006api.c into the application project.

The application source file must include tdrv006api.h. If these steps are done, the TDRV006 API can be used and the devices will be accessible.

3 API Documentation

3.1 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.

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 */
}
```

RETURN VALUE

A device descriptor pointer or NULL if the function fails.

3.2 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 the device
*/
result = tdrv006Close(hdl);
if (result != TDRV006_OK)
{
    /* handle close error */
}
```

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 specified device handle is invalid

3.3 tdrv006Read

NAME

tdrv006Read – Read state of I/O lines

SYNOPSIS

```
TDRV006_STATUS tdrv006Read
(
        TDRV006_HANDLE          hdl,
        unsigned int            *pIn31_0,
        unsigned int            *pIn63_32
)
```

DESCRIPTION

This function reads the current state of the I/O lines of the specified TDRV006.

PARAMETER

hdl

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

pIn31_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.

pIn63_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;
unsigned int       in_low;
unsigned int       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

3.4 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 argument 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, reset 16-63)
*/
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

3.5 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 argument 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 current 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 current 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.

3.6 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

3.7 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

3.8 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

3.9 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. A timeout value of '-1' specifies that the function will never timeout.

EXAMPLE

```
#include “tdrv006api.h”

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Wait for a low-to-high transition on input line 0
** Timeout after 10 seconds
*/
result = tdrv006WaitForLowToHigh (hdl, 0, 10000);
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.

3.10 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. The timeout is specified in milliseconds. A timeout value of '-1' specifies that the function will never timeout.

EXAMPLE

```
#include “tdrv006api.h”

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Wait for a high-to-low transition on input line 0
** Timeout after 10 seconds
*/
result = tdrv006WaitForHighToLow (hdl, 0, 10000);
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.

3.11tdrv006WaitForAnyTrans

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 a transition. If the specified time has passed the function will return with an error. The timeout is specified in milliseconds. A timeout value of '-1' specifies that the function will never timeout.

EXAMPLE

```
#include “tdrv006api.h”

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

/*
** Wait for a transition on input line 0
** Timeout after 10 seconds
*/
result = tdrv006WaitForAnyTrans (hdl, 0, 10000);
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.

4 Appendix

4.1 Example Application

The example application shall give an overview about the use of the TDRV006 devices and how to use the TDRV006 API.

The example application is designed as an interactive console application, so make sure to properly redirect the standard input and standard output for the example application's address space. If using a Dynamic Download Build e.g. in a telnet shell, use the following command:

```
# run -filtered <example_filename> -args <example_address_space>
```