The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV006-SW-95

QNX6 - Neutrino Device Driver

64 Digital Inputs/Outputs (Bit I/O)

Version 1.0.x

User Manual

Issue 1.0.0

September 2009

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
nfo@tews.com www.tews.com

TDRV006-SW-95

QNX6 - Neutrino Device Driver

64 Digital Inputs/Outputs (Bit I/O)

Supported Modules:
TPMC681

Issue	Description	Date
1.0.0	First Issue	September 9, 2009

Table of Contents

1 INTRODUCTION 4

2 INSTALLATION 5

2.1 Build the device driver 5

2.2 Build the example application 6

2.3 Start the driver process 6

3 DEVICE INPUT/OUTPUT FUNCTIONS 7

3.1 open() 7

3.2 close() 9

3.3 devctl() 10

3.3.1 DCMD_TDRV006_READ 12

3.3.2 DCMD_TDRV006_WRITE_MASK 14

3.3.3 DCMD_TDRV006_CONFIGURE_DIRECTION 16

3.3.4 DCMD_TDRV006_READ_DIRECTION 18

3.3.5 DCMD_TDRV006_OUTPUTBIT_SET 20

3.3.6 DCMD_TDRV006_OUTPUTBIT_CLEAR 21

3.3.7 DCMD_TDRV006_EVENT_WAIT 22

4 API DOCUMENTATION 24

4.1 General Functions 24

4.1.1 tdrv006open() 24

4.1.2 tdrv006close() 26

4.2 Device Access Functions 28

4.2.1 tdrv006read() 28

4.2.2 tdrv006write() 30

4.2.3 tdrv006writeMask() 32

4.2.4 tdrv006configDir() 34

4.2.5 tdrv006outputLinesSet() 36

4.2.6 tdrv006outputLinesClear() 38

4.2.7 tdrv006setOutputLine() 40

4.2.8 tdrv006clearOutputLine() 42

4.2.9 tdrv006eventWait() 44

1 Introduction

The TDRV006-SW-95 QNX-Neutrino device driver allows the operation of TDRV006 Digital I/O devices on QNX-Neutrino operating systems.

The TDRV006 device driver is basically implemented as a user installable Resource Manager. The standard file (I/O) functions (open, close and devctl) provide the basic interface for opening and closing a file descriptor and for performing device I/O and control operations.

The TDRV006-SW-95 device driver supports the following features:

- Reading from input buffers
- Writing to output buffers
- Configuring I/O line directions
- Waiting for several input event types

The TDRV006-SW-95 device driver supports the modules listed below:

TPMC681	64 Digital Inputs/Outputs (Bit 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 TDRV006 devices it is recommended to read the manuals listed below.

TPMC681 User manual

TPMC681 Engineering Manual

2 Installation

Following files are located on the distribution media:

Directory path 'TDRV006-SW-95':

TDRV006-SW-95-SRC.tar.gz	GZIP compressed archive with driver source code
TDRV006-SW-95-1.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

The GZIP compressed archive TDRV006-SW-95-SRC.tar.gz contains the following files and directories:

Directory path 'tdrv006':

driver/tdrv006.c	Driver source code
driver/tdrv006.h	Definitions and data structures for driver and application
driver/tdrv006def.h	Device driver include
driver/node.c	Queue management source code
driver/node.h	Queue management definitions
driver/nto/*	Build path
api/tdrv006api.h	API include file for application
api/tdrv006api.c	API source file
example/tdrv006exa.c	Example application
example/nto/*	Build path

For installation copy the tar-archive into the /usr/src directory and unpack it (e.g. `tar -xzvf TDRV006-SW-95-SRC.tar.gz`). After that the necessary directory structure for the automatic build and the source files are available underneath the new directory called *tdrv006*.

It is absolutely important to extract the TDRV006 tar archive in the /usr/src directory. Otherwise the automatic build with make will fail.

2.1 Build the device driver

Change to the /usr/src/tdrv006/driver directory

Execute the Makefile:

```
# make install
```

After successful completion the driver binary (tdrv006) will be installed in the /bin directory.

2.2 Build the example application

Change to the /usr/src/tdrv006/example directory.

Make sure that either symbolic links to the TDRV006 Application Programming Interface source and header file (tdrv006api.c and tdrv006api.h) located in */usr/src/tdrv006/api/* are available in the example's directory, or simply copy the API files into the directory. The API must be compiled together with the provided example applications.

Execute the Makefile:

```
# make install
```

After successful completion the example binary (***tdrv006exa***) will be installed in the /bin directory.

2.3 Start the driver process

To start the TDRV006 device driver, you have to enter the process name with optional parameter from the command shell or in the startup script.

```
tdrv006 [-v] &
```

The TDRV006 Resource Manager creates one device for each supported module, and registers the created devices in the Neutrinos pathname space under following names.

```
/dev/tdrv006_0
/dev/tdrv006_1
…
/dev/tdrv006_x
```

The reference between the created device names and the physical devices depends on the search order of the QNX Neutrino PCI bus driver.

The pathname must be used in the application program to open a path to the desired TDRV006 device.

For debugging purposes, you can start the TDRV006 Resource Manager with the –v option. Now the Resource Manager will print versatile information about TDRV006 configuration and command execution on the terminal window.

```
tdrv006 –v &
```

Make sure that only one instance of the device driver process is started.

3 Device Input/Output functions

This chapter describes the interface to the device driver I/O system.

3.1 open()

NAME

open() - open a file descriptor

SYNOPSIS

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open (const char *pathname, int flags)

DESCRIPTION

The *open* function creates and returns a new file descriptor for the TDRV006 device named by pathname. The flags argument controls how the file is to be opened. TDRV006 devices must be opened O_RDWR.

EXAMPLE

```
int fd;

fd = open("/dev/tdrv006_0", O_RDWR);
if (fd == -1)
{
     /* Handle error */
}
```

RETURNS

The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of –1 is returned. The global variable errno contains the detailed error code.

ERRORS

Returns only Neutrino specific error codes, see Neutrino Library Reference.

SEE ALSO

Library Reference - open()

3.2 close()

NAME

close() – close a file descriptor

SYNOPSIS

#include <unistd.h>

int close (int filedes)

DESCRIPTION

The close function closes the file descriptor *filedes*.

EXAMPLE

```
int fd;

...

if (close(fd) != 0)
{
     /* handle close error conditions */
}
```

RETURNS

The normal return value from close is 0. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

Returns only Neutrino specific error code, see Neutrino Library Reference.

SEE ALSO

Library Reference - close()

3.3 devctl()

NAME

devctl() – device control functions

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>
#include <devctl.h>

int devctl
(
        int             filedes,
        int             dcmd,
        void            *data_ptr,
        size_t          n_bytes,
        int             *dev_info_ptr
)
```

DESCRIPTION

The devctl function sends a control code directly to a device, specified by *filedes*, causing the corresponding device to perform the requested operation.

The argument *dcmd* specifies the control code for the operation.

The arguments *data_ptr* and *n_bytes* depends on the command and will be described for each command in detail later in this chapter. Usually *data_ptr* points to a buffer that passes data between the user task and the driver and *n_bytes* defines the size of this buffer.

The argument *dev_info_ptr* is unused for the TDRV006 driver and should be set to NULL.

The following devctl command codes are defined in *tdrv006.h*:

Value	Description
DCMD_TDRV006_READ	Read value from input buffer
DCMD_TDRV006_WRITE_MASK	Write new output value with bitmask
DCMD_TDRV006_CONFIGURE_DIRECTION	Configure direction of I/O lines
DCMD_TDRV006_READ_DIRECTION	Read current configuration of I/O direction
DCMD_TDRV006_OUTPUTBIT_SET	Set individual output line
DCMD_TDRV006_OUTPUTBIT_CLEAR	Clear individual output line
DCMD_TDRV006_EVENT_WAIT	Wait for an input event

See behind for more detailed information on each control code.

To use these TDRV006 specific control codes the header file tdrv006.h must be included in the application.

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERRORS

Returns only Neutrino specific error codes, see Neutrino Library Reference.

Other function dependent error codes will be described for each devctl code separately. Note, the TDRV006 driver always returns standard QNX error codes.

SEE ALSO

Library Reference - devctl()

3.3.1 DCMD_TDRV006_READ

NAME

DCMD_TDRV006_READ – Read value from input buffer

DESCRIPTION

This function reads the module input buffer, including input and output values. A pointer to the callers data buffer (TDRV006_RW_BUFFER) and the size of this structure are passed by the parameters data_ptr and n_bytes to the device.

```
typedef struct {
        unsigned int        value_31_0;
        unsigned int        value_63_32;
} TDRV006_RW_BUFFER;
```

value_31_0

This value returns the state of the I/O lines 0...31. Bit 0 contains the state of I/O line 0, bit 1 contains the state of line 1 and so on.

value_63_32

This value returns the state of the I/O lines 32...63. Bit 0 contains the state of I/O line 32, bit 1 contains the state of line 33 and so on.

EXAMPLE

```
#include "tdrv006.h"

int                 fd;
int                 result;
TDRV006_RW_BUFFER   ReadBuf;


result = devctl(    fd,
                    DCMD_TDRV006_READ,
                    &ReadBuf,
                    sizeof(TDRV006_RW_BUFFER),
                    NULL);

if (result == EOK) {
     printf( "Value: %08X %08X\n",
                    ReadBuf.value_63_32,
                    ReadBuf.value_31_0);
```

```
} else {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small.

3.3.2 DCMD_TDRV006_WRITE_MASK

NAME

DCMD_TDRV006_WRITE_MASK – Write new output value with bitmask

DESCRIPTION

This function writes a specified set of data bits to the digital output register. A pointer to the callers data buffer (TDRV006_RW_MASKED_BUFFER) and the size of this structure are passed by the parameters data_ptr and n_bytes to the device.

```
typedef struct {
        unsigned int        value_31_0;
        unsigned int        value_63_32;
        unsigned int        mask_31_0;
        unsigned int        mask_63_32;
} TDRV006_RW_MASKED_BUFFER;
```

value_31_0

This value specifies the output value for the lines 0...31. Bit 0 specifies the value for line 0, bit 1 specifies the value for line 1 and so on.

value_63_32

This value specifies the output value for the lines 32...63. Bit 0 specifies the value for line 32, bit 1 specifies the value for line 33 and so on.

mask_31_0

This parameter specifies the bitmask for the lines 0...31. Only active bits (1) will be written to the output registers. Bit 0 masks the value of line 0, bit 1 masks the value of line 1 and so on.

mask_63_32

This parameter specifies the bitmask for the lines 32...63. Only active bits (1) will be written to the output registers. Bit 0 masks the value of line 32, bit 1 masks the value of line 33 and so on.

EXAMPLE

```
#include "tdrv006.h"

int                          fd;
int                          result;
TDRV006_RW_MASKED_BUFFER     WriteBuf;

/*
** Write new output value
**   Change only lines 0..7, 15, 16 and 31
*/
WriteBuf.value_31_0     = 0x12345678;
WriteBuf.value_63_32    = 0xFFFFFFFF;
WriteBuf.mask_31_0      = 0x800180FF;
WriteBuf.mask_63_32     = 0x00000000; /* no lines to modify   */


result = devctl(   fd,
                   DCMD_TDRV006_WRITE_MASK,
                   &WriteBuf,
                   sizeof(TDRV006_RW_MASKED_BUFFER),
                   NULL);

if (result != EOK) {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small.

3.3.3 DCMD_TDRV006_CONFIGURE_DIRECTION

NAME

DCMD_TDRV006_CONFIGURE_DIRECTION – Configure direction of I/O lines

DESCRIPTION

This function configures the I/O line direction for input or output. A pointer to the callers data buffer (TDRV006_RW_ BUFFER) and the size of this structure are passed by the parameters data_ptr and n_bytes to the device.

```
typedef struct {
        unsigned int        value_31_0;
        unsigned int        value_63_32;
} TDRV006_RW_BUFFER;
```

value_31_0

This value specifies the direction for the lines 0...31. Bit 0 specifies the direction for line 0, bit 1 specifies the direction for line 1 and so on. A set bit (1) will enable output. A cleared bit (0) configures the I/O line for input.

value_63_32

This value specifies the direction for the lines 32...63. Bit 0 specifies the direction for line 32, bit 1 specifies the direction for line 33 and so on. A set bit (1) will enable output. A cleared bit (0) configures the I/O line for input

EXAMPLE

```
#include "tdrv006.h"

int                 fd;
int                 result;
TDRV006_RW_BUFFER   DirBuf;

/*
** Configure I/O line 0..15 for input and 56..63 for output
*/
DirBuf.value_31_0  = 0x0000FFFF;
DirBuf.value_63_32 = 0xFF000000;

...
```

```
result = devctl(    fd,
                    DCMD_TDRV006_CONFIGURE_DIRECTION,
                    &DirBuf,
                    sizeof(TDRV006_RW_BUFFER),
                    NULL);

if (result != EOK) {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small.

3.3.4 DCMD_TDRV006_READ_DIRECTION

NAME

DCMD_TDRV006_READ_DIRECTION – Read current configuration of I/O direction

DESCRIPTION

This function reads the current direction configuration of the I/O line direction. A pointer to the callers data buffer (TDRV006_RW_ BUFFER) and the size of this structure are passed by the parameters data_ptr and n_bytes to the device.

```
typedef struct {
        unsigned int        value_31_0;
        unsigned int        value_63_32;
} TDRV006_RW_BUFFER;
```

value_31_0

This value holds the direction configuration for the lines 0...31. Bit 0 corresponds to line 0, bit 1 corresponds to line 1 and so on. A set bit (1) represents an output configuration. A cleared bit (0) represents an input configuration.

value_63_32

This value holds the direction configuration for the lines 32...63. Bit 0 corresponds to line 32, bit 1 corresponds to line 33 and so on. A set bit (1) represents an output configuration. A cleared bit (0) represents an input configuration.

EXAMPLE

```
#include "tdrv006.h"

int                 fd;
int                 result;
TDRV006_RW_BUFFER   DirBuf;

/*
** Read current I/O line direction configuration
*/
result = devctl(    fd,
                    DCMD_TDRV006_READ_DIRECTION,
                    &DirBuf,
                    sizeof(TDRV006_RW_BUFFER),
                    NULL);
...
```

```
if (result == EOK) {
     printf( "Direction (1=output, 0=input): %08X %08X\n",
                    DirBuf.value_31_0,
                    DirBuf.value_63_32);

} else {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small.

3.3.5 DCMD_TDRV006_OUTPUTBIT_SET

NAME

DCMD_TDRV006_OUTPUTBIT_SET – Set individual output line

DESCRIPTION

This function sets a single output line leaving all other output lines in the current state. A pointer to the line number data buffer (int) and the size of this parameter are passed by the parameters data_ptr and n_bytes to the device. Valid values for this parameter are 0 to 63.

EXAMPLE

```
#include "tdrv006.h"

int             fd;
int             result;
int             lineNo;

/*
** Set output line 42 to HIGH
*/
lineNo = 42;
result = devctl(   fd,
                   DCMD_TDRV006_OUTPUTBIT_SET,
                   &lineNo,
                   sizeof(int),
                   NULL);

if (result != EOK) {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small, or an invalid parameter has been specified.

3.3.6 DCMD_TDRV006_OUTPUTBIT_CLEAR

NAME

DCMD_TDRV006_OUTPUTBIT_CLEAR – Clear individual output line

DESCRIPTION

This function clears a single output line leaving all other output lines in the current state. A pointer to the line number data buffer (int) and the size of this parameter are passed by the parameters data_ptr and n_bytes to the device. Valid values for this parameter are 0 to 63.

EXAMPLE

```
#include "tdrv006.h"

int             fd;
int             result;
int             lineNo;

/*
** Clear output line 0 to LOW
*/
lineNo = 0;
result = devctl(    fd,
                    DCMD_TDRV006_OUTPUTBIT_CLEAR,
                    &lineNo,
                    sizeof(int),
                    NULL);

if (result != EOK) {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small, or an invalid parameter has been specified.

3.3.7 DCMD_TDRV006_EVENT_WAIT

NAME

DCMD_TDRV006_EVENT_WAIT – Wait for an input event

DESCRIPTION

This function waits for a specific event (rising or falling edge) on a specific input line. A pointer to the callers data buffer (TDRV006_EVENT_BUFFER) and the size of this structure are passed by the parameters data_ptr and n_bytes to the device.

```
typedef struct {
        int     mode;
        int     inputLine;
        long    timeout;
} TDRV006_EVENT_BUFFER;
```

mode

This parameter specifies the event mode for this request. Following values are possible:

Value	Description
TDRV006_HIGH_TR	The driver waits for a high-transition at the specified input line.
TDRV006_LOW_TR	The driver waits for a low-transition at the specified input line.
TDRV006_ANY_TR	The driver waits for a high- or low-transition at the specified input line.

inputLine

This parameter specifies the input line number, which should be used to wait for the described event. Possible values are 0 to 63.

timeout

Specifies the amount of time (in seconds) the caller is at least willing to wait for the specified event to occur. A value of -1 means wait indefinitely or no timeout.

EXAMPLE

```
#include "tdrv006.h"

int                     fd;
int                     result;
TDRV006_EVENT_BUFFER    EventBuf;

/*
** Wait up to 20 seconds for a HIGH transition at input line 12
*/
EventBuf.mode           = TDRV006_HIGH_TR;
EventBuf.intputLine     = 12;
EventBuf.timeout        = 20;

result = devctl(   fd,
                   DCMD_TDRV006_EVENT_WAIT,
                   &EventBuf,
                   sizeof(TDRV006_EVENT_BUFFER),
                   NULL);

if (result == EOK) {
     printf("Event occurred.\n");
} else {
     /* process devctl() error */
}
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the data buffer is too small, or an invalid parameter has been specified.
ETIMEDOUT	The specified time to wait for the event has elapsed.

4 API Documentation

4.1 General Functions

4.1.1 tdrv006open()

Name

tdrv006open() – opens a device.

Synopsis

```
int 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.

Parameters

DeviceName

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

Example

```
#include "tdrv006api.h"
int FileDescriptor;

/*
** open file descriptor to device
*/
FileDescriptor = tdrv006open( "/dev/tdrv006_0" );
if (FileDescriptor < 0)
{
          /* handle open error */
}
```

RETURNS

A device descriptor number, or -1 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.

4.1.2 tdrv006close()

Name

tdrv006close() – closes a device.

Synopsis

```
int tdrv006close
(
     int  FileDescriptor
);
```

Description

This function closes previously opened devices.

Parameters

FileDescriptor

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

Example

```
#include "tdrv006api.h"
int FileDescriptor;
int result;

/*
** close file descriptor to device
*/
result = tdrv006close( FileDescriptor );
if (result < 0)
{
     /* handle close error */
}
```

RETURNS

EOK, or -1 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.

4.2 Device Access Functions

4.2.1 tdrv006read()

Name

tdrv006read() – read current input value.

Synopsis

```
int tdrv006read
(
     int                FileDescriptor,
     unsigned int*      pInputValue_31_0,
     unsigned int*      pInputValue_63_32
);
```

Description

This function reads the current value of the input register of the specified device.

Parameters

FileDescriptor

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

pInputValue_31_0

This value is a pointer to an unsigned int buffer which receives the current value of the input register for I/O lines 0…31. Bit 0 of this value corresponds to I/O line 0, bit 1 corresponds to I/O line 1 and so on. A NULL pointer can be specified, if the input value is not needed.

pInputValue_63_32

This value is a pointer to an unsigned int buffer which receives the current value of the input register for I/O lines 32…63. Bit 0 of this value corresponds to I/O line 32, bit 1 corresponds to I/O line 33 and so on. A NULL pointer can be specified, if the input value is not needed.

Example

```
#include "tdrv006api.h"

int                 FileDescriptor;
int                 result;
unsigned int        inVal_31_0;
unsigned int        inVal_63_32;

/*
** read current input value
*/
result = tdrv006read(   FileDescriptor,
                        &inVal_31_0,        /* Input 0..31     */
                        &inVal_63_32 );     /* Input 32..63    */
if (result == EOK)
{
     printf( "Input Value: %08X  %08X\n",
               inVal_63_32, inVal_31_0);
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

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

4.2.2 tdrv006write()

Name

tdrv006write() – write new output value.

Synopsis

```
int tdrv006write
(
     int                FileDescriptor,
     unsigned int       OutputValue_31_0,
     unsigned int       OutputValue_63_32
);
```

Description

This function writes a new output value for the specified device. All output lines are affected, and will be set according to the specified output values.

Parameters

FileDescriptor

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

OutputValue_31_0

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

OutputValue_63_32

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

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** write new output value:
** set I/O line 0 and I/O line 17 to ON, all other lines to OFF.
*/
result = tdrv006write(
              FileDescriptor,
              0x00020001,        /* OutputValue 0..31        */
              0x00000000);       /* OutputValue 32..63       */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

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

4.2.3 tdrv006writeMask()

Name

tdrv006writeMask() – write new output value with bitmask

Synopsis

```
int tdrv006writeMask
(
     int                FileDescriptor,
     unsigned int       OutputValue_31_0,
     unsigned int       OutputValue_63_32,
     unsigned int       Mask_31_0,
     unsigned int       Mask_63_32
);
```

Description

This function writes a specified set of output lines for the specified device. Only specified output lines are affected, others will be left unchanged.

Parameters

FileDescriptor

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

OutputValue_31_0

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

OutputValue_63_32

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

mask_31_0

This parameter specifies the bitmask for the lines 0...31. Only active bits (1) will be written to the output registers. Bit 0 masks the value of line 0, bit 1 masks the value of line 1 and so on.

mask_63_32

This parameter specifies the bitmask for the lines 32...63. Only active bits (1) will be written to the output registers. Bit 0 masks the value of line 32, bit 1 masks the value of line 33 and so on.

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Write new output value
**   Change only lines 0..7, 15, 16 and 31
*/
result = tdrv006writeMask(
              FileDescriptor,
              0x12345678,        /* OutputValue 0..31        */
              0x9ABCDEF0,        /* OutputValue 32..63       */
              0x800180FF,        /* Mask 0..31               */
              0x00000000 );      /* Mask 32..63              */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

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

4.2.4 tdrv006configDir()

Name

tdrv006configDir() – Configure direction of I/O lines

Synopsis

```
int tdrv006configDir
(
     int                FileDescriptor,
     unsigned int       Direction_31_0,
     unsigned int       Direction_63_32
);
```

Description

This function configures the I/O direction for the specified device. The direction of all I/O lines will be affected.

Parameters

FileDescriptor

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

Direction_31_0

This value specifies the direction for the lines 0...31. Bit 0 specifies the direction for line 0, bit 1 specifies the direction for line 1 and so on. A set bit (1) will enable output. A cleared bit (0) configures the I/O line for input.

Direction_63_32

This value specifies the direction for the lines 32...63. Bit 0 specifies the direction for line 32, bit 1 specifies the direction for line 33 and so on. A set bit (1) will enable output. A cleared bit (0) configures the I/O line for input

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Configure I/O line 0..15 for input and 56..63 for output
*/
DirBuf.value_31_0  = 0x0000FFFF;
DirBuf.value_63_32 = 0xFF000000;
result = tdrv006writeMask(
              FileDescriptor,
              0x0000FFFF,        /* Direction 0..31          */
              0xFF000000 );      /* Direction 32..63         */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

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

4.2.5 tdrv006outputLinesSet()

Name

tdrv006outputLinesSet() – Set specific output lines.

Synopsis

```
int tdrv006outputLinesSet
(
     int                FileDescriptor,
     unsigned int       Mask_31_0,
     unsigned int       Mask_63_32
);
```

Description

This function sets specified output lines to HIGH. Other output lines are left unchanged.

Parameters

FileDescriptor

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

Mask_31_0

Set (1) bits of this value will be set on the output lines 0...31. Cleared (0) bits will be left unchanged. Bit 0 of this value corresponds to I/O line 0, bit 1 corresponds to I/O line 1 and so on.

Mask_63_32

Set (1) bits of this value will be set on the output lines 32...63. Cleared (0) bits will be left unchanged. Bit 0 of this value corresponds to I/O line 32, bit 1 corresponds to I/O line 33 and so on.

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Set specific output lines:
** set I/O line 0 and I/O line 17 to ON, leave other lines unchanged.
*/
result = tdrv006outputLinesSet(
              FileDescriptor,
              0x00020001,        /* Output 0..31         */
              0x00000000);       /* Output 32..63        */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

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

4.2.6 tdrv006outputLinesClear()

Name

tdrv006outputLinesClear() – clear specific output lines.

Synopsis

```
int tdrv006outputLinesClear
(
     int                FileDescriptor,
     unsigned int       Mask_31_0,
     unsigned int       Mask_63_32
);
```

Description

This function clears specified output lines to LOW. Other output lines are left unchanged.

Parameters

FileDescriptor

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

Mask_31_0

Set (1) bits of this value will be cleared on the output lines 0...31. Cleared (0) bits will be left unchanged. Bit 0 of this value corresponds to I/O line 0, bit 1 corresponds to I/O line 1 and so on.

Mask_63_32

Set (1) bits of this value will be cleared on the output lines 32...63. Cleared (0) bits will be left unchanged. Bit 0 of this value corresponds to I/O line 32, bit 1 corresponds to I/O line 33 and so on.

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Clear specific output lines:
** Clear I/O lines 0..7, leave other lines unchanged.
*/
result = tdrv006outputLinesSet(
              FileDescriptor,
              0x000000FF,        /* Output 0..31         */
              0x00000000);       /* Output 32..63        */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

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

4.2.7 tdrv006setOutputLine()

Name

tdrv006setOutputLine() – set a specific output line.

Synopsis

```
int tdrv006setOutputLine
(
     int                FileDescriptor,
     int                outputLine
);
```

Description

This function sets the specified output line to HIGH. All other output lines are left unchanged.

Parameters

FileDescriptor

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

outputLine

This value specifies the I/O line which shall be set. Valid values are between 0 and 63.

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Set I/O line 42, leave other lines unchanged.
*/
result = tdrv006setOutputLine(
              FileDescriptor,
              42);               /* Output Line Number       */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

EINVAL	Invalid argument. This error code is returned if the specified output line number is out of range.

4.2.8 tdrv006clearOutputLine()

Name

tdrv006clearOutputLine() – clear a specific output line.

Synopsis

```
int tdrv006clearOutputLine
(
     int                FileDescriptor,
     int                outputLine
);
```

Description

This function clears the specified output line to LOW. All other output lines are left unchanged.

Parameters

FileDescriptor

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

outputLine

This value specifies the I/O line which shall be cleared. Valid values are between 0 and 63.

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Clear I/O line 42, leave other lines unchanged.
*/
result = tdrv006clearOutputLine(
              FileDescriptor,
              42);               /* Output Line Number       */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

EINVAL	Invalid argument. This error code is returned if the specified output line number is out of range.

4.2.9 tdrv006eventWait()

Name

tdrv006eventWait() – Wait for an input event.

Synopsis

```
int tdrv006eventWait
(
     int                FileDescriptor,
     int                mode,
     int                inputLine,
     long               timeout
);
```

Description

This function clears the specified output line to LOW. All other output lines are left unchanged.

Parameters

FileDescriptor

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

mode

This parameter specifies the event mode for this request. Following values are possible:

Value	Description
TDRV006_HIGH_TR	The driver waits for a high-transition at the specified input line.
TDRV006_LOW_TR	The driver waits for a low-transition at the specified input line.
TDRV006_ANY_TR	The driver waits for a high- or low-transition at the specified input line.

inputLine

This parameter specifies the input line number, which should be used to wait for the described event. Possible values are 0 to 63.

timeout

Specifies the amount of time (in seconds) the caller is at least willing to wait for the specified event to occur. A value of -1 means wait indefinitely or no timeout.

Example

```
#include "tdrv006api.h"

int           FileDescriptor;
int           result;

/*
** Wait up to 20 seconds for a HIGH transition at input line 12
*/
result = tdrv006eventWait(
              FileDescriptor,
              TDRV006_HIGH_TR,        /* Event type      */
              12,                     /* input line      */
              20);                    /* Timeout         */
if (result == EOK)
{
     /* OK */
}
else
{
     /* handle error */
}
```

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERROR CODES

EINVAL	Invalid argument. This error code is returned if the specified output line number is out of range.
ETIMEDOUT	The specified time to wait for the event has elapsed.