The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV003-SW-82

Linux Device Driver

16 (8) Bit Digital I/O

Version 3.0.x

User Manual

Issue 3.0.0

March 2024

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

TDRV003-SW-82

Linux Device Driver

16 (8) Bit Digital I/O

Supported Modules:
TPMC670
TPMC671

Issue	Description	Date
1.0	First Issue	August 21, 2000
1.1	Kernel 2.4.4 Support	August 23, 2001
1.2	General Revision	February 27, 2004
1.3.0	Kernel 2.6 Support	March 10, 2005
2.0.0	TPMC680-SW-82 changed to TDRV003-SW-82 TPMC671 Support added File list changed	August 8, 2006
2.0.1	New address TEWS LLC	August 29, 2006
2.0.2	New file list (config.h added), description of archive extraction	September 26, 2007
2.1.0	Description of new output functions added	June 12, 2008
2.1.1	Address TEWS LLC removed	July 20, 2010
2.1.2	Layout specific modifications	February 11, 2011
2.1.3	File-List modified	November 24, 2017
3.0.0	API functions implemented, Description of Ioctl functions removed, New Address TEWS Technologies GmbH	March 15, 2024

Table of Contents

1 INTRODUCTION 4

2 INSTALLATION 5

2.1 Build and install the Device Driver 5

2.2 Uninstall the Device Driver 6

2.3 Install the Device Driver into a running Kernel 6

2.4 Remove the Device Driver from a running Kernel 6

2.5 Change Major Device Number 7

3 API DOCUMENTATION 8

3.1 General Functions 8

3.1.1 tdrv003Open 8

3.1.2 tdrv003Close 10

3.2 Device Access Functions 12

3.2.1 tdrv003InputRead 12

3.2.2 tdrv003OutputWrite 14

3.2.3 tdrv003OutputRead 16

3.2.4 tdrv003OutputWriteMask 18

3.2.5 tdrv003OutputSetBits 20

3.2.6 tdrv003OutputClearBits 22

3.2.7 tdrv003EventWait 24

3.2.8 tdrv003DebouncerEnable 26

3.2.9 tdrv003DebouncerDisable 28

3.2.10 tdrv003WatchdogEnable 30

3.2.11 tdrv003WatchdogDisable 32

3.2.12 tdrv003WatchdogReset 34

4 DIAGNOSTIC 36

1 Introduction

The TDRV003-SW-82 Linux device driver allows the operation of supported hardware conforming to the Linux I/O system specification.

The TDRV003-SW-82 device driver supports the following features:

- write new output value
- write new output value with mask
- set/clear individual output lines
- read state of input lines
- wait for interrupt events (rising/falling edge) on each input line
- start and stop the output watchdog
- acknowledge watchdog errors
- configure & start and stop input debouncing

The TDRV003-SW-82 device driver supports the modules listed below:

TPMC670	16 (8) Channel Digital I/O	(PMC)
TPMC671	16 Channel Digital I/O	(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 usage of TDRV003 devices it is recommended to read the manuals listed below.

TPMC670/TPMC671 User Manual

2 Installation

Following files are located on the distribution media:

Directory path 'TDRV003-SW-82':

TDRV003-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
TDRV003-SW-82-3.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

The GZIP compressed archive TDRV003-SW-82-SRC.tar.gz contains the following files and directories:

Directory path './tdrv003/':

tdrv003.c	TDRV003 device driver source
tdrv003def.h	TDRV003 driver include file
tdrv003.h	TDRV003 include file for driver and application
Makefile	Device Driver Makefile
makenode	Script for Device Node Creation in File System
api/tdrv003api.c	API source file
api/tdrv003api.h	API include file
include/tpxxxhwdep.c	Hardware dependent library
include/tpxxxhwdep.h	Hardware dependent library header file
include/tpmodule.c	Driver independent library
include/tpmodule.h	Driver independent library header file
include/config.h	Driver independent library header file
example/tdrv003exa.c	Example Application
example/Makefile	Makefile for Example Application
COPYING	Copy of the GNU Public License (GPL)

In order to perform an installation, extract all files of the archive TDRV003-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TDRV003-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tdrv003.h and api/tdrv003api.h to */usr/include*

2.1 Build and install the Device Driver

- Login as *root*
- Change to the target directory
- To create and install the driver in the module directory */lib/modules/<version>/misc* enter:

  # make install

- To update the device driver's module dependencies, enter:

  # **depmod -aq**

2.2 Uninstall the Device Driver

- Login as *root*
- Change to the target directory
- To remove the driver from the module directory */lib/modules/<version>/misc* enter:

  # make uninstall

2.3 Install the Device Driver into a running Kernel

- To load the device driver into the running kernel, login as root and execute the following commands:

  # modprobe tdrv003drv

- After the first build or if you are using dynamic major device allocation it is necessary to create new device nodes on the file system. Please execute the script file *makenode* to do this. If your kernel has enabled a device file system (devfs or sysfs with udev) then you have to skip running the *makenode* script. Instead of creating device nodes from the script the driver itself takes creating and destroying of device nodes in its responsibility.

  # sh makenode

On success the device driver will create a minor device for each TDRV003 module found. The first TDRV003 module can be accessed with device node /dev/tdrv003_0, the second module with device node /dev/tdrv003_1, and so on.

The assignment of device nodes to physical TDRV003 modules depends on the search order of the PCI bus driver.

2.4 Remove the Device Driver from a running Kernel

- To remove the device driver from the running kernel login as root and execute the following command:

  # modprobe -r tdrv003drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tdrv003_x nodes will be automatically removed from your file system after this.

Make sure that the driver isn't opened by any application program. If opened you will get the response "*tdrv003drv: Device or resource busy*" and the driver will still remain in the system until you close all opened files and execute *modprobe –r* again.

2.5 Change Major Device Number

This paragraph is only for Linux kernels without DEVFS installed. The TDRV003 driver use dynamic allocation of major device numbers per default. If this isn't suitable for the application it is possible to define a major number for the driver.

To change the major number edit the file tdrv003def.h, change the following symbol to appropriate value and enter *make install* to create a new driver.

TDRV003_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TDRV003_MAJOR        122
```

Be sure that the desired major number is not used by other drivers. Please check */proc/devices* to see which numbers are free.

Keep in mind that it is necessary to create new device nodes if the major number for the TDRV003 driver has changed and the makenode script is not used.

3 API Documentation

3.1 General Functions

3.1.1 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 /dev/tdrv003_0, the second /dev/tdrv003_1, and so on.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tdrv003Open(“/dev/tdrv003_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.*

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

3.1.2 tdrv003Close

NAME

tdrv003Close – Closes a Device

SYNOPSIS

```
TDRV003_STATUS tdrv003Close
(
     TDRV003_HANDLE              hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE hdl;
TDRV003_STATUS result;

/*
** close file descriptor to device
*/
result = tdrv003Close( hdl );

if (result != TDRV003_OK)
{
     /* handle close error */
}
```

RETURNS

On success TDRV003_OK, or an appropriate error code.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

3.2 Device Access Functions

3.2.1 tdrv003InputRead

NAME

tdrv003InputRead – read state of input lines

SYNOPSIS

```
TDRV003_STATUS tdrv003InputRead
(
    TDRV003_HANDLE          hdl,
    unsigned short          *pInputBuf
)
```

DESCRIPTION

This function reads the current state of the input lines.

PARAMETERS

hdl

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

pInputBuf

This argument points to a buffer where the value will be returned.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;
unsigned short   data;


result = tdrv003InputRead(hdl, &data);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

3.2.2 tdrv003OutputWrite

NAME

tdrv003OutputWrite – write a new value to the output port

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputWrite
(
      TDRV003_HANDLE      hdl,
      unsigned short      OutputValue
)
```

DESCRIPTION

This function writes a new value to the output port.

PARAMETERS

hdl

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

OutputValue

This argument specifies the new output value. Bit 0 specifies the new state of output line 1; bit 1 specifies the new state for output line 2 and so on.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;

// set OUTPUT1 and OUTPUT16 and clear all other
result = tdrv003OutputWrite(hdl, 0x8001);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

Other returned error codes are system error conditions.

3.2.3 tdrv003OutputRead

NAME

tdrv003OutputRead – read current state of output lines

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputRead
(
        TDRV003_HANDLE          hdl,
        unsigned short          *pOutputBuf
)
```

DESCRIPTION

This function reads the current state of the output lines.

PARAMETERS

hdl

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

pOutputBuf

This argument points to a buffer where the value will be returned.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;
unsigned short   data;


result = tdrv003OutputRead(hdl, &data);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

Other returned error codes are system error conditions.

3.2.4 tdrv003OutputWriteMask

NAME

tdrv003OutputWriteMask – writes a masked value to the output port

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputWriteMask
(
    TDRV003_HANDLE      hdl,
    unsigned short      OutputValue,
    unsigned short      mask
)
```

DESCRIPTION

This control function writes a masked value to the output port. Only those bits in value for which the corresponding bit position in the mask is set to 1 will be changed in the output port.

PARAMETERS

hdl

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

OutputValue

This argument specifies the masked value for the output port. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on.

mask

This argument specifies the mask for relevant bits. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on.
Only those bits in *OutputValue* for which the corresponding bit position in this mask is set to 1 will be changed in the output port.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE  hdl;
TDRV003_STATUS  result;

// clear OUTPUT1 and set OUTPUT16 and leave all other lines unchanged
result = tdrv003OutputWriteMask(hdl, 0x8000, 0x8001);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

Other returned error codes are system error conditions.

3.2.5 tdrv003OutputSetBits

NAME

tdrv003OutputSetBits – set specific output lines

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputSetBits
(
      TDRV003_HANDLE          hdl,
      unsigned short          OutputBits
)
```

DESCRIPTION

This function sets specific bits in the output port to active state (1).

PARAMETERS

hdl

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

OutputBits

This argument specifies a mask of relevant bits to set. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on. Bits which are set (1) in this mask will be set in the corresponding bits in the output port.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;

// set OUTPUT1 and OUTPUT16 to active state
result = tdrv003OutputSetBits(hdl, (1<<15) | (1<<0));

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

Other returned error codes are system error conditions.

3.2.6 tdrv003OutputClearBits

NAME

tdrv003OutputClearBits – clear specific output lines

SYNOPSIS

```
TDRV003_STATUS tdrv003OutputClearBits
(
      TDRV003_HANDLE          hdl,
      unsigned short          OutputBits
)
```

DESCRIPTION

This function clears specific bits in the output port to inactive state (0).

PARAMETERS

hdl

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

OutputBits

This argument specifies a mask of relevant bits to clear. Bit 0 corresponds to the first output line; bit 1 corresponds to the second output line and so on. Bits which are set (1) in this mask will be cleared in the corresponding bits in the output port.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;

// clear OUTPUT1 and OUTPUT16
result = tdrv003OutputSetBits(hdl, (1<<15) | (1<<0));

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.
TDRV003_ERR_IO	The output register is locked by a watchdog failure. Execute the function tdrv003WatchdogReset to reset the watchdog error.

Other returned error codes are system error conditions.

3.2.7 tdrv003EventWait

NAME

tdrv003EventWait – wait for a specific input event

SYNOPSIS

```
TDRV003_STATUS tdrv003EventWait
(
    TDRV003_HANDLE      hdl,
    unsigned short      mode,
    unsigned short      mask,
    long                timeout
)
```

DESCRIPTION

This function waits for an event (rising or falling edge or both) at the specified input lines. The function is blocked until at least one of the specified events or a timeout occurs.

PARAMETERS

hdl

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

mode

This argument specifies the kind of event to wait for. The following event types are defined in tdrv003.h:

Event Type	Description
TDRV003_HIGH_TR	Wait for a low to high transition on a specified input line.
TDRV003_LOW_TR	Wait for a high to low transition on a specified input line.
TDRV003_ANY_TR	Wait for any transition on a specified input line.

mask

This argument specifies a bit mask of input lines to wait for the specified edge event. Bit 0 corresponds to the first input line; bit 1 corresponds to the second input line and so on.
Only one bit shall be set in the mask, otherwise the occurred event cannot be determined exactly. If more than one bit is set in the mask the function is completed the moment a relevant transition at least at one specified bit position occurs.

timeout

Specifies the amount of time (in 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 value of MAX_REQUESTS in tdrv003def.h.
TDRV003_ERR_TIMEOUT	The requested event does not occur within the specified time (timeout).

Other returned error codes are system error conditions.

3.2.8 tdrv003DebouncerEnable

NAME

tdrv003DebouncerEnable – configure and enable debouncer circuit

SYNOPSIS

```
TDRV003_STATUS tdrv003DebouncerEnable
(
    TDRV003_HANDLE      hdl,
    unsigned short      DebounceTimer
)
```

DESCRIPTION

This function configures and enables the input debouncer circuit.

PARAMETERS

hdl

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

DebounceTimer

This argument specifies the debouncer time. The debouncer time is specified in approx. 7µs steps. Please refer to the corresponding Hardware User Manual for a calculation formula and a table of values.

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;

// Enable the debouncer with a debounce time of 1ms
result = tdrv003DebouncerEnable(hdl, 147);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

3.2.9 tdrv003DebouncerDisable

NAME

tdrv003DebouncerDisable – disable debouncer circuit

SYNOPSIS

```
TDRV003_STATUS tdrv003DebouncerDisable
(
     TDRV003_HANDLE              hdl
)
```

DESCRIPTION

This function disables the input debouncer circuit.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Disable the debouncer circuit
result = tdrv003DebouncerDisable(hdl);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

3.2.10 tdrv003WatchdogEnable

NAME

tdrv003WatchdogEnable – enable output watchdog

SYNOPSIS

```
TDRV003_STATUS tdrv003WatchdogEnable
(
     TDRV003_HANDLE              hdl
)
```

DESCRIPTION

This function enables the watchdog timer for the output lines. The watchdog function is activated after the next write operation to the device. Please remember that if the watchdog is enabled and no write access occurs within 120 ms, all outputs go into the inactive (0) state. To unlock the output register and leave the inactive state the function *tdrv003WatchdogReset* must be executed.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Enable output watchdog
result = tdrv003WatchdogEnable(hdl);

if (result != TDRV003_OK)
{
     /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

3.2.11 tdrv003WatchdogDisable

NAME

tdrv003WatchdogDisable – disable output watchdog

SYNOPSIS

```
TDRV003_STATUS tdrv003WatchdogDisable
(
    TDRV003_HANDLE          hdl
)
```

DESCRIPTION

This function disables the watchdog timer for the output lines.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Disable output watchdog
result = tdrv003WatchdogDisable(hdl);

if (result != TDRV003_OK)
{
    /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

3.2.12 tdrv003WatchdogReset

NAME

tdrv003WatchdogReset – reset output watchdog error

SYNOPSIS

```
TDRV003_STATUS tdrv003WatchdogReset
(
    TDRV003_HANDLE          hdl
)
```

DESCRIPTION

This device function resets an output watchdog error. This function must be called after a device function returns the error code *TDRV003_ERR_IO.*

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv003api.h”

TDRV003_HANDLE   hdl;
TDRV003_STATUS   result;


// Reset watchdog error
result = tdrv003WatchdogReset (hdl);

if (result != TDRV003_OK)
{
    /* handle error */
}
```

RETURNS

On success, TDRV003_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV003_ERR_INVALID_HANDLE	The specified TDRV003_HANDLE is invalid.

Other returned error codes are system error conditions.

4 Diagnostic

If the TDRV003 does not work properly it is helpful to get some status information from the driver respective the kernel.

The Linux */proc* file system provides information about kernel, resources, driver, devices, and so on. The following screen dumps display information of a correct running TDRV003 driver (see also the proc man pages). The examples show a system with 1x TPMC671.

```
# lspci -v
…

03:01.0 Signal processing controller: TEWS Technologies GmbH Device 029f
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: slow devsel, IRQ 19
        Memory at a1100000 (32-bit, non-prefetchable) [size=128]
        I/O ports at 4000 [size=128]
        I/O ports at 4080 [size=16]
        Kernel driver in use: TEWS Technologies TDRV003 16(8) Digital IO
        Kernel modules: tdrv003drv
…

# cat /proc/devices
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 ttyS
  5 cua
  6 lp
  7 vcs
 10 misc
 13 input
 29 fb
 36 netlink
129 ptm
…
142 pts
162 raw
254 tdrv003drv
```

```
# cat /proc/interrupts
           CPU0
  0:     282100          XT-PIC  timer
  1:          6          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  8:          1          XT-PIC  rtc
 11:       4648          XT-PIC  eth0
 12:        223          XT-PIC  PS/2 Mouse
 14:      12968          XT-PIC  ide0
 15:          0          XT-PIC  ide1
 19:          8          XT-PIC  TDRV003
NMI:          0
ERR:          0

# cat /proc/ioports
0000-0cf7 : PCI Bus 0000:00
  0000-001f : dma1
  0020-0021 : pic1
  0040-0043 : timer0
  0050-0053 : timer1
  0060-0060 : keyboard
  0064-0064 : keyboard
  0070-0071 : rtc_cmos
    0070-0071 : rtc0
[…]
0d00-ffff : PCI Bus 0000:00
  1800-1803 : ACPI PM1a_EVT_BLK
  1804-1805 : ACPI PM1a_CNT_BLK
  1808-180b : ACPI PM_TMR
  1850-1850 : ACPI PM2_CNT_BLK
  1854-1857 : pnp 00:04
  1880-189f : ACPI GPE0_BLK
  2000-20fe : pnp 00:06
  3000-3fff : PCI Bus 0000:04
    3000-30ff : 0000:04:00.0
  4000-4fff : PCI Bus 0000:02
    4000-4fff : PCI Bus 0000:03
      4000-407f : 0000:03:01.0
      4080-408f : 0000:03:01.0
        4080-408f : TDRV003
  5000-503f : 0000:00:02.0
[…]
```