The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC600-SW-82

Linux Device Driver

32/16 Digital Inputs (24V)

Version 2.0.x

User Manual

Issue 2.0.0

June 2020

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
Phone: +49 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com www.tews.com

TPMC600-SW-82

Linux Device Driver

32/16 Digital Inputs (24V)

Supported Modules:
TPMC600

This document contains information, which is proprietary to TEWS TECHNOLOGIES GmbH. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES GmbH reserves the right to change the product described in this document at any time without notice.

TEWS TECHNOLOGIES GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2009-2020 by TEWS TECHNOLOGIES GmbH

Issue	Description	Date
1.0.0	First Issue	September 16, 2009
1.0.1	Layout specific modifications	February 12, 2011
2.0.0	API implemented, description of ioctl removed	June 24,.2020

Table of Contents

1 Introduction

The TPMC600-SW-82 Linux device driver allows the operation of the TPMC600 Digital Input PMC conforming to the Linux I/O system specification

The TPMC600-SW-82 device driver supports the following features:

- Read digital input value immediately or after a selected event occurs
- Wait for events on input lines
- Configure input hardware debouncing

The TPMC600-SW-82 device driver supports the modules listed below:

TPMC600-10	32 digital inputs (Front Panel I/O)	(PMC)
TPMC600-11	16 digital inputs (Front Panel I/O)	(PMC)
TPMC600-20	32 digital inputs (P14 I/O)	(PMC)
TPMC600-21	16 digital inputs (P14 I/O)	(PMC)

To get more information about the features and use of TPMC600 devices it is recommended to read the manuals listed below.

TPMC600 User Manual

2 Installation

The directory TPMC600-SW-82 on the distribution media contains the following files:

TPMC600-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
TPMC600-SW-82-2.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

The GZIP compressed archive TPMC600-SW-82-SRC.tar.gz contains the following files and directories:

Directory path 'tpmc600':

tpmc600.c	TPMC600 device driver source
tpmc600def.h	TPMC600 driver include file
tpmc600.h	TPMC600 include file for driver and application
Makefile	Device Driver Makefile
makenode	Script for Device Node Creation in File System
COPYING	Copy of the GNU Public License (GPL)
api/tpmc600api.h	API include file
api/tpmc600api.c	API source 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/tpmc600exa.c	Example Application
example/Makefile	Makefile for Example Application

In order to perform an installation, extract all files of the archive TPMC600-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC600-SW-82-SRC.tar.gz' will extract the files into the local directory.

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 tpmc600drv

- 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 TPMC600 module found. The first TPMC600 module can be accessed with device node /dev/tpmc600_0, the second module with device node /dev/tpmc600_1, and so on.

The assignment of device nodes to physical TPMC600 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 tpmc600drv

If your kernel has enabled a dynamic file system, all /dev/tpmc600_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 "*tpmc600drv: 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 TPMC600 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 tpmc600def.h, change the following symbol to appropriate value and enter *make install* to create a new driver.

TPMC600_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC600_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.

2.6 Maximum Number of Active Jobs Configuration

The maximum number of active event read jobs per major device can be configured. This can be simply made by changing the value of the symbol in tpmc600def.h.

NUM_REQUESTS	Defines the maximum number of active event wait jobs (default = 100). Valid numbers are in range between 1 and MAXINT.

3 API Documentation

3.1 General Functions

3.1.1 tpmc600Open

NAME

tpmc600Open – open a device

SYNOPSIS

```
TPMC600_HANDLE tpmc600Open
(
        char            *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle must be opened by a call to this function.

The tpmc600Open function can be called multiple times (e.g. in different tasks).

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TPMC600 device is named “/dev/tpmc600_0” the second device is named “/dev/tpmc600_1” and so on.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tpmc600Open("/dev/tpmc600_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.

3.1.2 tpmc600Close

NAME

tpmc600Close – close a device

SYNOPSIS

```
TPMC600_STATUS tpmc600Close
(
      TPMC600_HANDLE        hdl
)
```

DESCRIPTION

This function closes a previously opened device.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE      hdl;
TPMC600_STATUS      result;

/*
** close the device
*/
result = tpmc600Close(hdl);
if (result != TPMC600_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 tpmc600GetPciInfo

NAME

tpmc600GetPciInfo – get PCI information of the module

SYNOPSIS

```
TPMC600_STATUS tpmc600GetPciInfo
(
    TPMC600_HANDLE          hdl,
    TPMC600_PCIINFO_BUF     *pPciInfoBuf
)
```

DESCRIPTION

This function returns information about the module's PCI header as well as the PCI localization.

PARAMETERS

hdl

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

pPciInfoBuf

This argument is a pointer to the structure TPMC600_PCIINFO_BUF that receives information of the module PCI header.

```
typedef struct
{
    unsigned short      vendorId;
    unsigned short      deviceId;
    unsigned short      subSystemId;
    unsigned short      subSystemVendorId;
    int                 pciBusNo;
    int                 pciDevNo;
    int                 pciFuncNo;
} TPMC600_PCIINFO_BUF;
```

vendorId
PCI module vendor ID.

deviceId
PCI module device ID

subSystemId
PCI module sub system ID

subSystemVendorId
PCI module sub system vendor ID

pciBusNo
Number of the PCI bus, where the module resides.

pciDevNo
PCI device number

pciFuncNo
PCI function number

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;
TPMC600_PCIINFO_BUF     pciInfoBuf;

/*
** get module PCI information
*/
result = tpmc600GetPciInfo( hdl, &pciInfoBuf );
if (result == TPMC600_OK)
{
     printf( "PCI Localization (Bus:Dev.Func): %d:%d.%d\n",
                    pciInfoBuf.pciBusNo,
                    pciInfoBuf.pciDevNo,
                    pciInfoBuf.pciFuncNo );
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_INVAL	Specified pointer is invalid.

3.2 Device Access Functions

3.2.1 tpmc600Read

NAME

tpmc600Read – read input state of device

SYNOPSIS

```
TPMC600_STATUS tpmc600Read
(
    TPMC600_HANDLE          hdl,
    unsigned int            *pDigInVal
)
```

DESCRIPTION

This function reads the current input state of the device.

PARAMETERS

hdl

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

pDigInVal

This argument points to a 32-bit buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;
unsigned int       in_value;

/*
** read current state of I/O lines
*/
result = tpmc600Read(hdl, &in_value);
if (result == TPMC600_OK)
{
     printf("input value: 0x%08X\n", in_value);
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_INVAL	Specified pointer is invalid.

3.2.2 tpmc600EnableDebouncer

NAME

tpmc600EnableDebouncer – configure and enable input debouncer

SYNOPSIS

```
TPMC600_STATUS tpmc600EnableDebouncer
(
    TPMC600_HANDLE          hdl,
    unsigned short          debValue
)
```

DESCRIPTION

This function configures the input debouncer, which shall prevent the board to detect fast faulty signal changes.

PARAMETERS

hdl

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

debValue

This argument specifies the debouncer timer value. Valid values are 0 for 7us and 65535 for 440ms. Please refer to the TPMC600 User Manual for a detailed description.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;

/*
** enable debouncer with a debounce time of ~1ms (143)
*/
result = tpmc600EnableDebouncer(hdl, 143);
if (result == TPMC600_OK)
{
     /* debouncer enabled */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid

3.2.3 tpmc600DisableDebouncer

NAME

tpmc600DisableDebouncer – disable input debouncer

SYNOPSIS

```
TPMC600_STATUS tpmc600DisableDebouncer
(
      TPMC600_HANDLE              hdl
)
```

DESCRIPTION

This function disables the input debouncer.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;

/*
** disable debouncer function
*/
result = tpmc600DisableDebouncer(hdl);
if (result == TPMC600_OK)
{
     /* debouncer disabled */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid

3.2.4 tpmc600WaitForAnyEvent

NAME

tpmc600WaitForAnyEvent – wait for transition on input line

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForAnyEvent
(
    TPMC600_HANDLE      hdl,
    int                 inputLine,
    int                 msTimeout
)
```

DESCRIPTION

This function waits for a transition on the specified input line.

PARAMETERS

hdl

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

inputLine

This argument specifies the input line. A value of 1 must be specified for IN1, 2 for IN2 and so on. The maximum valid input line depends on the used module variant.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;

/*
** wait for a transition (any) on IN12
**     timeout after approx. 10 sec.
*/
result = tpmc600WaitForAnyEvent(hdl, 12, 10000);
if (result == TPMC600_OK)
{
     /* event occurred */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_TIMEOUT	The event did not occur in the specified time

3.2.5 tpmc600WaitForHighEvent

NAME

tpmc600WaitForHighEvent – wait for low-to-high transition on input line

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForHighEvent
(
    TPMC600_HANDLE      hdl,
    int                 inputLine,
    int                 msTimeout
)
```

DESCRIPTION

This function waits for a low-to-high transition on the specified input line.

PARAMETERS

hdl

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

inputLine

This argument specifies the input line. A value of 1 must be specified for IN1, 2 for IN2 and so on. The maximum valid input line depends on the used module variant.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;

/*
** wait for a transition (low-to-high) on IN12
**     timeout after approx. 10 sec.
*/
result = tpmc600WaitForHighEvent(hdl, 12, 10000);
if (result == TPMC600_OK)
{
     /* event occurred */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_TIMEOUT	The event did not occur in the specified time

3.2.6 tpmc600WaitForLowEvent

NAME

tpmc600WaitForLowEvent – wait for high-to-low transition on input line

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForLowEvent
(
    TPMC600_HANDLE      hdl,
    int                 inputLine,
    int                 msTimeout
)
```

DESCRIPTION

This function waits for high-to-low transition on the specified input line.

PARAMETERS

hdl

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

inputLine

This argument specifies the input line. A value of 1 must be specified for IN1, 2 for IN2 and so on. The maximum valid input line depends on the used module variant.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;

/*
** wait for a transition (High-to-low) on IN12
**     timeout after approx. 10 sec.
*/
result = tpmc600WaitForLowEvent(hdl, 12, 10000);
if (result == TPMC600_OK)
{
     /* event occurred */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_TIMEOUT	The event did not occur in the specified time

3.2.7 tpmc600ReadOnAnyEvent

NAME

tpmc600ReadOnAnyEvent – wait for 1st transition on set of input lines and return input state

SYNOPSIS

```
TPMC600_STATUS tpmc600ReadOnAnyEvent
(
    TPMC600_HANDLE          hdl,
    unsigned int            inputMask,
    int                     msTimeout,
    unsigned int            *pDigInVal
)
```

DESCRIPTION

This function waits for the first transition on any of the specified input lines. After detection of the transition the input state will be read and returned.

There is a delay between the transition and actual reading of the input state. This means that the returned value represents the input state a short time after the transition has occurred.

PARAMETERS

hdl

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

inputMask

This argument specifies a mask of input lines that are active to wait for a transition. A set bit specifies an active input line. Input lines set to 0 will not be observed. Bit 0 is assigned to IN1, bit 1 to IN2, and so on.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;
unsigned int            inVal;

/*
** read input state after 1st transition (any)
**     on IN1, IN2, IN3, IN4, or IN16
**     timeout after approx. 5 sec.
*/
result = tpmc600ReadOnAnyEvent(hdl, 0x0000800F, 5000, &inVal);
if (result == TPMC600_OK)
{
     /* event occurred */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_TIMEOUT	The event did not occur in the specified time

3.2.8 tpmc600ReadOnHighEvent

NAME

tpmc600ReadOnHighEvent – wait for 1st low-to-high transition on set of input lines and return input state

SYNOPSIS

```
TPMC600_STATUS tpmc600ReadOnHighEvent
(
    TPMC600_HANDLE          hdl,
    unsigned int            inputMask,
    int                     msTimeout,
    unsigned int            *pDigInVal
)
```

DESCRIPTION

This function waits for the first low-to-high transition on any of the specified input lines. After detection of the transition the input state will be read and returned.

There is a delay between the transition and actual reading of the input state. This means that the returned value represents the input state a short time after the transition has occurred.

PARAMETERS

hdl

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

inputMask

This argument specifies a mask of input lines that are active to wait for a transition. A set bit specifies an active input line. Input lines set to 0 will not be observed. Bit 0 is assigned to IN1, bit 1 to IN2, and so on.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;
unsigned int            inVal;

/*
** read input state after 1st transition (low-to-high)
**     on IN1, IN2, IN3, IN4, or IN16
**     timeout after approx. 5 sec.
*/
result = tpmc600ReadOnHighEvent(hdl, 0x0000800F, 5000, &inVal);
if (result == TPMC600_OK)
{
     /* event occurred */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_TIMEOUT	The event did not occur in the specified time

3.2.9 tpmc600ReadOnLowEvent

NAME

tpmc600ReadOnLowEvent – wait for 1st high-to-low transition on set of input lines and return input state

SYNOPSIS

```
TPMC600_STATUS tpmc600ReadOnLowEvent
(
    TPMC600_HANDLE      hdl,
    unsigned int        inputMask,
    int                 msTimeout,
    unsigned int        *pDigInVal
)
```

DESCRIPTION

This function waits for the first high-to-low transition on any of the specified input lines. After detection of the transition the input state will be read and returned.

There is a delay between the transition and actual reading of the input state. This means that the returned value represents the input state a short time after the transition has occurred.

PARAMETERS

hdl

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

inputMask

This argument specifies a mask of input lines that are active to wait for a transition. A set bit specifies an active input line. Input lines set to 0 will not be observed. Bit 0 is assigned to IN1, bit 1 to IN2, and so on.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include "tpmc600api.h"

TPMC600_HANDLE          hdl;
TPMC600_STATUS          result;
unsigned int            inVal;

/*
** read input state after 1st transition (high-to-low)
**     on IN1, IN2, IN3, IN4, or IN16
**     timeout after approx. 5 sec.
*/
result = tpmc600ReadOnLowEvent(hdl, 0x0000800F, 5000, &inVal);
if (result == TPMC600_OK)
{
     /* event occurred */
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC600_ERR_TIMEOUT	The event did not occur in the specified time

4 Diagnostic

If the TPMC600 does not work properly it is helpful to get some status information from the driver respective kernel.

The Linux */proc* file system provides information about kernel, resources, drivers, devices and so on. The following screen dumps display information of a correct running TPMC600 device driver (see also the proc man pages).

```
# lspci -v
…
04:02.0 Signal processing controller: TEWS Technologies GmbH Device 0258
(rev 01)
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: medium devsel, IRQ 17
        Memory at feb9f400 (32-bit, non-prefetchable) [size=128]
        I/O ports at e800 [size=128]
        Memory at feb9f000 (32-bit, non-prefetchable) [size=32]
        Kernel driver in use: TEWS TECHNOLOGIES TPMC600 16(8) Digital IO
        Kernel modules: tpmc600drv
```

```
# cat /proc/devices
Character devices:
  1 mem
  …
248 tpmc600drv
  …
```

```
# cat /proc/ioports
0000-0cf7 : PCI Bus 0000:00
  0000-001f : dma1
  0020-0021 : pic1
  …
  e000-efff : PCI Bus 0000:04
    e800-e87f : 0000:04:02.0
    e880-e8ff : 0000:04:01.0
    ec00-ec3f : 0000:04:00.0
  …
```

```
# cat /proc/iomem
00000000-00000fff : Reserved
00001000-0009fbff : System RAM
0009fc00-0009ffff : Reserved
…
  feb00000-febfffff : PCI Bus 0000:04
    feb9f000-feb9f01f : 0000:04:02.0
    feb9f400-feb9f47f : 0000:04:02.0
    feba0000-febbffff : 0000:04:00.0
    febc0000-febdffff : 0000:04:00.0
…


# cat /proc/interrupts
           CPU0       …       CPUn
  0:         107       …          0   IO-APIC   2-edge      timer
  1:           0       …         11   IO-APIC   1-edge      i8042
  8:           1       …          0   IO-APIC   8-edge      rtc0
  9:           0       …          0   IO-APIC   9-fasteoi   acpi
  …
 17:           0       …          0   IO-APIC  17-fasteoi   TPMC600
```