The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV006-SW-82

Linux Device Driver

64 Digital Inputs/Outputs (Bit I/O)

Version 2.1.x

User Manual

Issue 2.1.0

June 2022

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-82

Linux Device Driver

64 (32) Digital Inputs/Outputs (Bit I/O)

Supported Modules:
TPMC681
TPMC321

Issue	Description	Date
1.0.0	First Issue	December 20, 2005
1.0.1	File list changed	August 15, 2006
1.0.2	New Address TEWS LLC	January 17, 2007
1.0.3	Address TEWS LLC removed	June 24, 2010
1.0.4	General Revision	May 4, 2011
1.1.0	Support of kernel versions 3.x	February 2, 2012
2.0.0	API implemented, General Revision	December 14, 2012
2.0.1	File-list modified	December 1, 2017
2.1.0	Support for TPMC321 added	June 9, 2022

Table of Contents

1 Introduction

The TDRV006-SW-82 Linux device driver allows the operation of the TDRV006 compatible PMCs conforming to the Linux I/O system specification.

The TDRV006-SW-82 device driver supports the following features:

- Reading input I/O value
- Writing to output buffers (masked and unmasked)
- Setting I/O bits individually
- Waiting for input transition events
- Configuring I/O direction

The TDRV006-SW-82 supports the modules listed below:

TPMC321-10	64 Digital Inputs / Outputs (Bit I/O, TTL)	(PMC, Conduction Cooled)
TPMC321-11	32 Digital Inputs / Outputs (Bit I/O, EIA-422/EIA-485)	(PMC, Conduction Cooled)
TPMC321-12	32 Digital Inputs / Outputs (Bit I/O, M-LVDS)	(PMC, Conduction Cooled)
TPMC681-10	64 Digital Inputs / Outputs (Bit I/O, TTL)	(PMC)

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

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

TPMC681 User Manual

2 Installation

Following files are located on the distribution media:

Directory path 'TDRV006-SW-82':

TDRV006-SW-82-2.1.0.pdf	This manual in PDF format
TDRV006-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
ChangeLog.txt	Release history
Release.txt	Release information

For installation the files have to be copied to the desired target directory.

The GZIP compressed archive TDRV006-SW-82-SRC.tar.gz contains the following files and directories:

Directory path 'tdrv006':

tdrv006.c	Driver source code
tdrv006def.h	Driver include file
tdrv006.h	Driver include file for application program
makenode	Script to create device nodes on the file system
Makefile	Device driver make file
api/tdrv006api.c	API source file
api/tdrv006api.h	API include file
example/tdrv006exa.c	Example application
example/Makefile	Example application make file
include/config.h	Driver independent configuration header file
include/tpmodule.h	Driver and kernel independent library header file
include/tpmodule.c	Driver and kernel independent library source file
include/tpxxxhwdep.h	HAL library header file
include/tpxxxhwdep.c	HAL library source file
COPYING	Copy of the GNU Public License (GPL)

In order to perform an installation, extract all files of the archive TDRV006-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TDRV006-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tdrv006.h and api/tdrv006api.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 Device Driver into the running Kernel

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

  # modprobe tdrv006drv

- 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 compatible channel found. The first PMC module can be accessed with device node /dev/tdrv006_0, the second module with device node /dev/tdrv006_1 and so on.

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

2.4 Remove Device Driver from the running Kernel

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

  # modprobe -r tdrv006drv

If your kernel has enabled a device file system (devfs or sysfs with udev), all /dev/tdrv006_* nodes will be automatically removed from your file system after this.

Be sure that the driver is not opened by any application program. If opened you will get the response ``t*drv006drv: 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 dynamic device file system installed.

The TDRV006 driver uses dynamic allocation of major device numbers per default. If this isn't suitable for the application it's possible to define a major number for the driver.

To change the major number edit the file tdrv006def.h, change the following symbol to appropriate value and enter `make install` to create a new driver.

TDRV006_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TDRV006_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 TDRV006 driver has changed and the makenode script is not used.

2.6 Configuration

To adjust application specific driver properties see tdrv006def.h and look for the following symbol defines (#define <symbol> <value>):

TDRV006_MAX_EVENT_RECORDS

This symbol specifies the size of the interrupt routine event record queue. If you have input event loss during multiple Input Event jobs, please double the certain value.

TDRV006_MAX_EVENTWAIT_JOBS

This symbol specifies the maximum number of concurrent jobs waiting for Input Events.

3 API Documentation

3.1 General Functions

3.1.1 tdrv006Open

NAME

tdrv006Open – open a device.

SYNOPSIS

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

DESCRIPTION

Before I/O can be performed to a device, a device 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 TDRV006 device is named /dev/tdrv006_0, the second /dev/tdrv006_1, and so on.

EXAMPLE

```
#include <tdrv006api.h>

TDRV006_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tdrv006Open(“/dev/tdrv006_0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

The error codes are stored in *errno.*

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

3.1.2 tdrv006Close

NAME

tdrv006Close – close a device.

SYNOPSIS

```
TDRV006_STATUS tdrv006Close
(
    TDRV006_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 <tdrv006api.h>

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

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

RETURNS

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

ERROR CODES

Error Code	Description
TDRV006_ERR_INVALID_HANDLE	The specified device handle is invalid

3.2 Device Access Functions

3.2.1 tdrv006Read

NAME

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

SYNOPSIS

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

DESCRIPTION

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

PARAMETER

hdl

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

in31_0

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

in63_32

This argument points to a buffer where the current value of I/O lines 32 up to 63 will be returned. Bit 0 returns the value of I/O line 32, bit 1 the value of I/O line 33, and so on. (Returned value is only valid if I/O lines 32..63 are present)

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

3.2.2 tdrv006Write

NAME

tdrv006Write – set value of output buffer

SYNOPSIS

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

DESCRIPTION

This function sets the output value.

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

PARAMETER

hdl

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

out31_0

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

out63_32

This argument specifies the output value for I/O lines 32 up to 63. Bit 0 specifies the value of I/O line 32, bit 1 the value of I/O line 33, and so on. (Value is not used if I/O lines 32..63 are not present)

EXAMPLE

```
#include <tdrv006api.h>

TDRV006_HANDLE     hdl;
TDRV006_STATUS     result;

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

RETURN VALUE

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

ERROR CODES

Error Code	Description
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid

3.2.3 tdrv006WriteMasked

NAME

tdrv006WriteMasked – modify value of output buffer for specified I/O lines

SYNOPSIS

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

DESCRIPTION

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

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

PARAMETER

hdl

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

out31_0

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

out63_32

This argument specifies the output value for I/O lines 32 up to 63. Bit 0 specifies the value of I/O line 32, bit 1 the value of I/O line 33, and so on. (Value is not used if I/O lines 32..63 are not present)

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 I/O line shall be set to the value specified by *out31_0*.
A reset bit (0) means that the old output value will not be changed.

mask63_32

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

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

3.2.4 tdrv006SetOutputLine

NAME

tdrv006SetOutputLine – set a specified output line

SYNOPSIS

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

DESCRIPTION

This function sets a single bit of the output value.

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

PARAMETER

hdl

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

outputLine

This argument specifies a data bit that shall be set. Allowed values are 0 up to 63. (Not present I/O lines are ignored)

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

3.2.5 tdrv006ClearOutputLine

NAME

tdrv006ClearOutputLine – reset a specified I/O line

SYNOPSIS

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

DESCRIPTION

This function resets a single bit of the output value.

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

PARAMETER

hdl

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

outputLine

This argument specifies data bit that shall be reset. Allowed values are 0 up to 63. (Not present I/O lines are ignored)

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

3.2.6 tdrv006OutputEnable

NAME

tdrv006OutputEnable – set the direction of the I/O lines

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). (Not used if I/O lines 32..63 are not present)

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

3.3 Input Event Functions

3.3.1 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. (Only present I/O line will be supported)

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	The number of waiting processes exceeded the maximum number
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid
TDRV006_ERR_TIMEOUT	Timeout occurred.
TDRV006_ERR_NOSUPP	The specified I/O line does not exist

3.3.2 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	The number of waiting processes exceeded the maximum number
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid
TDRV006_ERR_TIMEOUT	Timeout occurred.
TDRV006_ERR_NOSUPP	The specified I/O line does not exist

3.3.3 tdrv006WaitForAnyTrans

NAME

tdrv006WaitForAnyTrans – wait until a transition occurs

SYNOPSIS

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

DESCRIPTION

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

PARAMETER

hdl

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

inputLine

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

timeout

This argument specifies the time the function is willing to wait for the specified transition. If the specified time has passed the function will return with an error. 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	The number of waiting processes exceeded the maximum number
TDRV006_ERR_INVALID_HANDLE	The device handle is invalid
TDRV006_ERR_TIMEOUT	Timeout occurred.
TDRV006_ERR_NOSUPP	The specified I/O line does not exist

4 Diagnostic

If the TDRV006 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, driver, devices and so on. The following screen dumps displays information of a correct running TDRV006 driver (see also the proc man pages).

```
# lspci -v
...
04:01.0 Signal processing controller: TEWS Technologies GmbH Device 02a9
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: medium devsel, IRQ 16
        Memory at feb9fc00 (32-bit, non-prefetchable) [size=128]
        I/O ports at e880 [size=128]
        Memory at feb9f800 (32-bit, non-prefetchable) [size=256]
        Kernel driver in use: TEWS TECHNOLOGIES TDRV006 64 Digital Bit IO
        Kernel modules: tdrv006drv
...

# cat /proc/devices
Character devices:
  1 mem
  2 pty
  3 ttyp
  4
  5 cua
  7 vcs
 10 misc
 13 input
 14 sound
 29 fb
 36 netlink
162 raw
180 usb
226 drm
254 tdrv006drv
```

```
# cat /proc/interrupts
  0:        125          0   IO-APIC-edge      timer
  1:          4          2   IO-APIC-edge      i8042
  7:          1          0   IO-APIC-edge      parport0
  8:          0          0   IO-APIC-edge      rtc0
  9:          0          0   IO-APIC-fasteoi   acpi
 12:         38         37   IO-APIC-edge      i8042
 14:         85         62   IO-APIC-edge      ata_piix
 15:          0          0   IO-APIC-edge      ata_piix
 16:          0          0   IO-APIC-fasteoi   uhci_hcd:usb5, TDRV006
 18:          0          0   IO-APIC-fasteoi   uhci_hcd:usb4
 19:          6        377   IO-APIC-fasteoi   uhci_hcd:usb3, p1p1
 22:       1444       1245   IO-APIC-fasteoi   ata_piix, i801_smbus
 23:          0          0   IO-APIC-fasteoi   ehci_hcd:usb1, uhci_hcd:usb2
 42:        354        383   PCI-MSI-edge      radeon
 43:         98         95   PCI-MSI-edge      snd_hda_intel
 44:         19         21   PCI-MSI-edge      snd_hda_intel
NMI:          3          3   Non-maskable interrupts
LOC:       9705       9516   Local timer interrupts
SPU:          0          0   Spurious interrupts
PMI:          3          3   Performance monitoring interrupts
…

# cat /proc/iomem

00000000-0000ffff : reserved
00010000-0009fbff : System RAM
0009fc00-0009ffff : reserved
000a0000-000bffff : PCI Bus 0000:00
000c0000-000cffff : Video ROM
000d0000-000dffff : PCI Bus 0000:00
  000d0000-000d0fff : Adapter ROM
…
80000000-ffffffff : PCI Bus 0000:00
  80000000-801fffff : PCI Bus 0000:03
  80200000-803fffff : PCI Bus 0000:03
…
  feb00000-febfffff : PCI Bus 0000:04
    feb9f800-feb9f8ff : 0000:04:01.0
      feb9f800-feb9f8ff : TDRV006
    feb9fc00-feb9fc7f : 0000:04:01.0
…
```