The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC501-SW-82

Linux Device Driver

32 Channel 16 Bit ADC

Version 2.0.x

User Manual

Issue 2.0.2

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

TPMC501-SW-82

Linux Device Driver

32 Channel 16 Bit ADC

Supported Modules:
TPMC501

Issue	Description	Date
1.0	First Issue	October 24, 2001
1.1	New ioctl() function TP501_IOCSMODTYPE	May 14, 2002
1.2	Distribution format has changed	December 17, 2003
1.3.0	Kernel 2.6.x Support	March 10, 2005
1.3.1	Filelist modified, New Address TEWS LLC, general revision	November 08, 2006
1.3.2	Installation description modified, read() parameter corrected	August 25, 2008
1.3.3	Address TEWS LLC removed	June 15, 2010
1.3.4	General Revision	February 8, 2012
2.0.0	General revision. API documentation added.	May 13, 2016
2.0.1	COPYING-File added to file-list	October 19, 2017
2.0.2	New Address TEWS Technologies GmbH	March 8, 2024

Table of Contents

1 Introduction

The TPMC501-SW-82 Linux device driver allows the operation of a TPMC501 ADC PMC on Linux operating systems.

The TPMC501-SW-82 device driver software includes the following features:

- read value from a selected ADC channel
- use sequencer mode for continuously reads from selected channels
- correction of input values with the factory programmed correction data
- select hardware supported gains

The TPMC501-SW-82 device driver supports the modules listed below:

TPMC501	Optically Isolated 32 Channel 12 Bit ADC	PMC

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

TPMC501 Hardware User Manual

2 Installation

Following files are located on the distribution media:

Directory path 'TPMC501-SW-82':

TPMC501-SW-82-2.0.2.pdf	This manual in PDF format
TPMC501-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
Release.txt	Release information
ChangeLog.txt	Release history

The GZIP compressed archive TPMC501-SW-82-SRC.tar.gz contains the following files and directories:

Directory path './tpmc501/':

tpmc501.c	Driver source code
tpmc501def.h	Driver include file
tpmc501.h	Driver include file for application program
Makefile	Device driver make file
makenode	Script for device node creation
COPYING	Copy of the GNU Public License (GPL)
api/tpmc501api.h	API include file
api/tpmc501api.c	API source file
include/config.h	Include of system dependent config.h
include/tpxxxhwdep.c	Low level hardware access functions source file
include/tpxxxhwdep.h	Access functions header file
include/tpmodule.c	Driver independent library
include/tpmodule.h	Driver independent library header file
example/tpmc501exa.c	Example application
example/Makefile	Example application makefile

In order to perform an installation, extract all files of the archive TPMC501-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC501-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tpmc501.h and api/tpmc501api.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 -a**

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 tpmc501drv

- 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 TPMC501 module found. The first module of the first TPMC501 module can be accessed with device node /dev/tpmc501_0, the second module with device node /dev/tpmc501_1, the third TPMC501 module with device node /dev/tpmc501_2 and so on.

The assignment of device nodes to physical TPMC501 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 tpmc501drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc501_x nodes will be automatically removed from your file system after this.

Be sure that the driver isn't opened by any application program. If opened you will get the response "*tpmc501drv: 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 management. The TPCM500 driver use 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 tpmc501def.h, change the following symbol to appropriate value and enter `make install` to create a new driver.

TPMC501_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC501_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 TPMC501 driver has changed and the makenode script is not used.

3 API Documentation

3.1 General Functions

3.1.1 tpmc501Open

NAME

tpmc501Open – Opens a Device

SYNOPSIS

```
TPMC501_HANDLE tpmc501Open
(
      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 TPMC501 device is named “/dev/tpmc501_0” the second device is named “/dev/tpmc501_1” and so on.

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tpmc501Open(“/dev/tpmc501_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 tpmc501Close

NAME

tpmc501Close – Closes a Device

SYNOPSIS

```
TPMC501_STATUS tpmc501Close
(
      TPMC501_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 “tpmc501api.h”

TPMC501_HANDLE hdl;
TPMC501_STATUS result;

/*
** close file descriptor to device
*/
result = tpmc501Close( hdl );

if (result != TPMC501_OK)
{
  /* handle close error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 tpmc501SetModelType

NAME

tpmc501SetModelType – Set the module type of the TPMC501

SYNOPSIS

```
TPMC501_STATUS tpmc501SetModelType
(
    TPMC501_HANDLE      hdl,
    int                 ModuleType
);
```

DESCRIPTION

This function configures the model type of the TPMC501.

This function must be called before the first AD conversion can be started.

PARAMETERS

hdl

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

ModuleType

This argument specifies the model type of the TPMC501. The following model types are supported.

Value	Description
TPMC501_TYPE_10	TPMC501-10 (Gain 1/2/5/10, +/-10V, Front I/O)
TPMC501_TYPE_11	TPMC501-11 (Gain 1/2/4/8, +/-10V, Front I/O)
TPMC501_TYPE_12	TPMC501-12 (Gain 1/2/5/10, 0-10V, Front I/O)
TPMC501_TYPE_13	TPMC501-13 (Gain 1/2/4/8, 0-10V, Front I/O)
TPMC501_TYPE_20	TPMC501-20 (Gain 1/2/5/10, +/-10V, Back I/O)
TPMC501_TYPE_21	TPMC501-21 (Gain 1/2/4/8, +/-10V, Back I/O)
TPMC501_TYPE_22	TPMC501-22 (Gain 1/2/5/10, 0-10V, Back I/O)
TPMC501_TYPE_23	TPMC501-23 (Gain 1/2/4/8, 0-10V, Back I/O)

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE   hdl;
TPMC501_STATUS   result;

result = tpmc501SetModelType(hdl, TPMC501_TYPE_11);

if (result != TPMC501_OK)
{
   /* handle error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.
TPMC501_ERR_INVAL	Invalid model type specified.

3.1.4 tpmc501GetModuleInfo

NAME

tpmc501GetModuleInfo – Get module information data

SYNOPSIS

```
TPMC501_STATUS tpmc501GetModuleInfo
(
  TPMC501_HANDLE       hdl,
  TPMC501_INFO_BUFFER  *pModuleInfo
);
```

DESCRIPTION

This function reads module information data such as configured module type, location on the PCI bus and factory programmed correction data.

PARAMETERS

hdl

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

pModuleInfo

This argument specifies a pointer to the module information buffer.

```
typedef struct
{
   unsigned int     Variant;
   unsigned int     PciBusNo;
   unsigned int     PciDevNo;
   int              ADCOffsetCal[4];
   int              ADCGainCal[4];
} TPMC501_INFO_BUFFER;
```

Variant

This parameter returns the configured module variant (e.g. 10 for a TPMC501-10).

PciBusNo, PciDevNo

These parameters specify the PCI location of this module.

ADCOffsetCal[4]

This array returns the factory programmed offset correction values for the different gain settings. Array index 0 contains the value for gain 1, index 1 contains the value for gain 2 and so forth.

ADCGainCal[4]

This array returns the factory programmed gain correction for the different gain settings. Array index 0 contains the value for gain 1, index 1 contains the value for gain 2 and so forth.

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE       hdl;
TPMC501_STATUS       result;
TPMC501_INFO_BUFFER  ModuleInfo

result = tpmc501GetModuleInfo(hdl, &ModuleInfo);

if (result != TPMC501_OK)
{
   /* handle error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.

3.2 Device Access Functions

3.2.1 tpmc501Read

NAME

tpmc501Read – Read converted AD value

SYNOPSIS

```
TPMC501_STATUS tpmc501Read
(
    TPMC501_HANDLE      hdl,
    int                 channel,
    int                 gain,
    int                 flags,
    int                 *pAdcVal
);
```

DESCRIPTION

This function starts an AD conversion on the specified channel and returns the converted value.

PARAMETERS

hdl

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

channel

This argument specifies the input channel number. Valid channels for single-ended mode are 1…32, for differential mode 1...16.

gain

This argument specifies the gain for this channel. Valid gains are 1, 2, 5, 10 for *TPMC501-10/-12/-20/-22* and 1, 2, 4, 8 for *TPMC501-11/-13/-21/-23.*

flags

Set of bit flags that control the AD conversion. The following flags could be OR'ed:

Flag	Meaning
TPMC501_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC501_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC501 EEPROM.
TPMC501_FAST	If this flag is set the fast (polled) mode will be used. The driver will not use interrupts, instead it will wait in a busy loop until the settling time (if necessary) and the conversion is finished. Conversions using this mode will be handled faster, but the processor executes a busy loop and other tasks will not be handled during the loops.

pAdcVal

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

EXAMPLE

```
#include "tpmc501api.h"

TPMC501_HANDLE   hdl;
TPMC501_STATUS   result;
int              AdcData;
int              channel, gain, flags;

channel   = 32;
gain      = 2;
flags     = TPMC501_CORR | TPMC501_FAST;

result = tpmc501Read(hdl, channel, gain, flags, &AdcData);

if (result != TPMC501_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_ACCESS	The module type has not been configured.
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.
TPMC501_ERR_INVAL	At least one of the parameters is invalid.
TPMC501_ERR_TIMEOUT	ADC conversion timed out.
TPMC501_ERR_RANGE	Invalid channel number.
TPMC501_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.2 tpmc501StartSequencer

NAME

tpmc501StartSequencer – Start sequencer operation

SYNOPSIS

```
TPMC501_STATUS tpmc501StartSequencer
(
    TPMC501_HANDLE      hdl,
    unsigned int        CycleTime,
    unsigned int        NumOfBufferPages,
    unsigned int        NumOfChannels,
    TPMC501_CHAN_CONF   *ChanConf
);
```

DESCRIPTION

This function sets up and starts the sequencer. The setup specifies the channels to be used in sequencer mode and how they will be setup, defining gain, correction and input interface. Additionally the sequencer cycle time is defined and the depth of the driver's sequencer FIFO will be configured.

PARAMETERS

hdl

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

CycleTime

This argument specifies the repeat frequency of the sequencer in 100 µs steps. Each time the sequencer timer reaches the programmed cycle time a new AD conversion of all active channels is started. Valid values are in the range from 100 microseconds to 6.5535 seconds.

NumOfBufferPages

This argument specifies the number of sample blocks in the ring buffer. A sample block contains the samples of all channels (NumOfChannels) per sequencer cycle.

NumOfChannels

This argument specifies the number of active channels for this job. The maximum number is 32.

ChanConf

This array of channel configuration structures specifies the configuration of the active channels. The channel configuration defines the channel number, the gain and some flags. The ordering of channels in a ring buffer page is the same as defined in this array.

```
typedef struct
{
        unsigned int        ChanToUse;
        unsigned int        gain;
        unsigned int        flags;
} TPMC501_CHAN_CONF;
```

ChanToUse

This parameter specifies the input channel number. Valid channels for single-ended mode are 1…32, for differential mode 1...16.

gain

This Parameter specifies the gain for this channel. Valid gains are 1, 2, 5, 10 for *TPMC501-10/-12/-20/-22* and 1, 2, 4, 8 for *TPMC501-11/-13/-21/-23.*

flags

Set of bit flags that control the AD conversion. The following flags could be OR'ed:

Flag	Meaning
TPMC501_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended mode (default).
TPMC501_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC501 EEPROM.

EXAMPLE

```
#include "tpmc501api.h"

TPMC501_HANDLE     hdl;
TPMC501_STATUS     result;
unsigned int       CycleTime;
unsigned int       NumOfBufferPages;
unsigned int       NumOfChannels;
TPMC501_CHAN_CONF  ChanConf[TPMC501_MAX_CHAN];

CycleTime          = 5000;
NumOfBufferPages   = 100;
NumOfChannels      = 2;

ChanConf[0].ChanToUse= 1;
ChanConf[0].gain     = 1;
ChanConf[0].flags    = TPMC501_CORR;

ChanConf[1].ChanToUse= 20;
ChanConf[1].gain     = 5;
ChanConf[1].flags    = TPMC501_CORR;
…
```

```
// start the sequencer
result = tpmc501StartSequencer(hdl, CycleTime, NumOfBufferPages,
                 NumOfChannels, ChanConf);

if (result != TPMC501_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_ACCESS	The module type has not been configured.
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.
TPMC501_ERR_INVAL	At least one of the parameters is invalid.
TPMC501_ERR_RANGE	Invalid channel number.
TPMC501_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.3 tpmc501GetDataBuffer

NAME

tpmc501GetDataBuffer – Get next data block of sequencer samples

SYNOPSIS

```
TPMC501_STATUS tpmc501GetDataBuffer
(
    TPMC501_HANDLE      hdl,
    unsigned int        flags,
    int                 *pData,
    unsigned int        *pStatus
)
```

DESCRIPTION

This function returns the next available data block in the ring buffer containing ADC data of configured sequencer channels.

If specified the function will return immediately, even if there is no data available. If the function should wait for data, the function returns immediately if data is already available in FIFO or waits for sequencer cycle completion. The function timeouts, if there is an abnormal delay during wait (sequencer cycle-time exceeded).

PARAMETERS

hdl

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

flags

Set of bit flags that control the sequencer read. The following flags could be OR'ed:

Flag	Meaning
TPMC501_NOWAIT	If this flag is set the function will return immediately, even if there is no data available. If the flag is not set, the function will wait until data is available.
TPMC501_FLUSH	If this flag is set the sequencer FIFO will be flushed and the function will wait for new data otherwise the function will read the next available data set.

pData

This argument is a pointer to an array of integer items where the converted data of a sequencer cycle will be filled in. The number of channels and the channel configuration was setup with the tpmc501StartSequencer function. The used buffer must be at least big enough to receive one integer value for every enabled sequencer channel.

The first array item [0] belongs to the channel configured by ChanConfig[0], the second array item [1] belongs to the channel configured by ChanConfig[1] and so forth. Please refer to the example application for details.

pStatus

This argument is a pointer to a variable which returns the sequencer error status. Keep in mind to check this status for each reading. If status is 0 no error is pending. A set of bits specifies the possible error conditions.

Value	Description
TPMC501_BUF_OVERRUN	This bit indicates a ring buffer overrun. The error occurred if there is no space in ring buffer to write the new AD data. In this case the new AD values are dismissed. The sequencer was not stopped.
TPMC501_DATA_OVERFLOW	This indicates an overrun in the sequencer data RAM. The error occurred if the driver is too slow to read the data in time. The sequencer was stopped after this error occurred.
TPMC501_TIMER_ERR	Sequencer timer error (see also TPMC501 hardware manual). The sequencer was stopped after this error occurred.
TPMC501_INST_RAM_ERR	Sequencer instruction RAM error (see also TPMC501 hardware manual). The sequencer was stopped after this error occurred.

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE   hdl;
TPMC501_STATUS   result;
unsigned int     seqStatus;
int              numOfSeqChannels;
int              *pData;

numOfSeqChannels = 2;   /* Two channels used in sequencer mode */

/* allocate sequencer input buffer */
pData = malloc(sizeof(int) * numOfSeqChannels);

…
```

…

```
/* read a set of fresh ADC data */
result = tpmc501GetDataBuffer(hdl, TPMC501_FLUSH, &pData, &seqStatus);
if (result != TPMC501_OK)
{
  /* handle error */
}
```

RETURN VALUE

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.
TPMC501_ERR_TIMEOUT	There the expected wait time has been exceeded.
TPMC501_ERR_NOT_READY	The sequencer is stopped.
TPMC501_ERR_NODATA	The function returned without data
TPMC501_ERR_BUSY	The device is not configured in sequencer mode

3.2.4 tpmc501StopSequencer

NAME

tpmc501StopSequencer – Stop the sequencer

SYNOPSIS

```
TPMC501_STATUS tpmc501StopSequencer
(
      TPMC501_HANDLE          hdl
);
```

DESCRIPTION

This function stops execution of the sequencer mode on the specified device.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE   hdl;
TPMC501_STATUS   result;

result = tpmc501StopSequencer(hdl);

if (result != TPMC501_OK)
{
   /* handle error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.

4 Diagnostic

If the TPMC501 does not work properly it is helpful to get some status information from the driver respective kernel. To get debug output from the driver enable the following symbols in '*tpmc501.c*' by replacing "#undef" with "#define":

```
#define DEBUG_TPMC501
```

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 TPMC501 driver (see also the proc man pages).

```
# tail –f /var/log/messages /* before modprobing the TPMC501 driver */
Jul 21 12:14:59 linux kernel: TEWS Technologies - TPMC501 32 Channel 12 Bit ADC
version 2.0.x (<Release Date>)

Jul 21 12:14:59 linux kernel: TPMC501:  Installing device (vendor=0x10B5,
device=0x9050) at 4:2.0

…

# lspci -v
…
04:02.0 Signal processing controller: PLX Technology, Inc. PCI <-> IOBus Bridge
(rev 01)
    Subsystem: TEWS Technologies GmbH Device 01f5
    Flags: medium devsel, IRQ 16
    Memory at feb9fc00 (32-bit, non-prefetchable) [size=128]
    I/O ports at e880 [size=128]
    I/O ports at e400 [size=256]
    Memory at feb9f000 (32-bit, non-prefetchable) [size=2K]
    Kernel driver in use: TEWS Technologies - TPMC501 32 Channel 16 Bit ADC
…

# cat /proc/devices
Character devices:
  1 mem
…
226 drm
253 tpmc501drv

Block devices:
  1 ramdisk
…
```

```
# cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
  0:         42          0          0          0   IO-APIC-edge      timer
  1:          4          3          1          2   IO-APIC-edge      i8042
  6:          1          1          1          0   IO-APIC-edge      floppy
  7:          1          0          0          0   IO-APIC-edge      parport0
  8:          0          0          1          0   IO-APIC-edge      rtc0
  9:          0          0          0          0   IO-APIC-fasteoi   acpi
 12:         39         42         44         38   IO-APIC-edge      i8042
 14:       1890         31         33       2735   IO-APIC-edge      ata_piix
 15:          0          0          0          0   IO-APIC-edge      ata_piix
 16:          0          0          0          0   IO-APIC-fasteoi   uhci_hcd:usb5
 17:         13          9          0         97   IO-APIC-fasteoi   TPMC501
 18:          0          0          0          0   IO-APIC-fasteoi   uhci_hcd:usb4
…

# cat /proc/ioports
…
  e000-efff : PCI Bus 0000:04
    e000-e0ff : 0000:04:02.0
    e400-e4ff : 0000:04:02.0
      e400-e4ff : TPMC501
    ec00-ec3f : 0000:04:00.0
      ec00-ec3f : e1000
  ffa0-ffaf : 0000:00:1f.1
    ffa0-ffaf : ata_piix
…
```