The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC550-SW-82

Linux Device Driver

8/4 Channels of Isolated 12 bit D/A

Version 1.0.x

User Manual

Issue 1.0.2

September 2010

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

TPMC550-SW-82

Linux Device Driver

8/4 Channels of Isolated 12 bit D/A

Supported Modules:
TPMC550

Issue	Description	Date
1.0.0	First Issue	December 8, 2006
1.0.1	File list modified	November 11, 2008
1.0.2	Address TEWS LLS removed	September 22, 2010

Table of Contents

1 Introduction

The TPMC550-SW-82 Linux device driver allows the operation of the TPMC550 PMC conforming to the Linux I/O system specification. This includes a device-independent basic I/O interface with *open()*, *close()* and *ioctl()* functions.

The TPMC550-SW-82 device driver supports the following features:

- write a new output value to a specified channel
- start and setup the output sequencer
- update sequencer output values
- stop the output sequencer
- read the module configuration

The TPMC550-SW-82 device driver supports the modules listed below:

TPMC550	8/4 Channels of Isolated 12 bit D/A	(PMC)

In this document all supported modules and devices will be called TPMC550. Specials for a certain devices will be advised.

To get more information about the features and use of TPMC550 devices it is recommended to read the manuals listed below.

TPMC550 User manual
TPMC550 Engineering Manual

2 Installation

The directory TPMC550-SW-82 on the distribution media contains the following files:

TPMC550-SW-82-1.0.2.pdf	This manual in PDF format
TPMC550-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
Release.txt	Release information
ChangeLog.txt	Release history

The GZIP compressed archive TPMC550-SW-82-SRC.tar.gz contains the following files and directories:

Directory path './tpmc550/':

tpmc550.c	Driver source code
tpmc550def.h	Driver private include file
tpmc550.h	Driver public include file for application program
Makefile	Device driver make file
makenode	Script to create device nodes on the file system
include/config.h	Driver independent library header file
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/tpmc550exa.c	Example application
example/Makefile	Example application make file

In order to perform an installation, extract all files of the archive TPMC550-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC550-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tpmc550.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

- Only after the first build we have to execute *depmod* to create a new dependency description for loadable kernel modules. This dependency file is later used by modprobe to load the driver module.

  # 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

- Update kernel module dependency description file

  # depmod -aq

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 tpmc550drv

- After the first build or if you are using dynamic major device allocation it's 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 TPMC550 module found. The first TPMC550 module can be accessed with device node /dev/tpmc550_0, the second with device node /dev/tpmc550_1 and so on.

The assignment of device nodes to physical TPMC550 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 tpmc550drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc550_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 "*tpmc550drv: 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 TPMC550 device driver uses 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 tpmc550def.h, change the following symbol to appropriate value and enter `make install` to create a new driver.

TPMC550_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC550_MAJOR       122
```

3 Device Input/Output functions

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

3.1 open()

NAME

open() - open a file descriptor

SYNOPSIS

#include <fcntl.h>

int **open** (*const char *filename, int flags*)

DESCRIPTION

The open function creates and returns a new file descriptor for the file named by *filename*. The *flags* argument controls how the file is to be opened. This is a bit mask; you create the value by the bitwise OR of the appropriate parameters (using the | operator in C).

See also the GNU C Library documentation for more information about the open function and open flags.

EXAMPLE

```
int fd;

fd = open(“/dev/tpmc550_0”, O_RDWR);
if (fd < 0)
{
     /* handle open error conditions */
}
```

RETURNS

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

ERRORS

E_NODEV	The requested minor device does not exist.

This is the only error code returned by the driver, other codes may be returned by the I/O system during open. For more information about open error codes, see the *GNU C Library description – Low-Level Input/Output*.

SEE ALSO

GNU C Library description – Low-Level Input/Output

3.2 close()

NAME

close() – close a file descriptor

SYNOPSIS

#include <unistd.h>

int **close** *(int filedes)*

DESCRIPTION

The close function closes the file descriptor *filedes*.

EXAMPLE

```
int fd;

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

RETURNS

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

ERRORS

E_NODEV	The requested minor device does not exist.

This is the only error code returned by the driver, other codes may be returned by the I/O system during close. For more information about close error codes, see the *GNU C Library description – Low-Level Input/Output*.

SEE ALSO

GNU C Library description – Low-Level Input/Output

3.3 ioctl()

NAME

ioctl() – device control functions

SYNOPSIS

#include <sys/ioctl.h>
#include <tpmc550.h>

int **ioctl**(int filedes, int request [, void *argp])

DESCRIPTION

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

The argument *request* specifies the control code for the operation. The optional argument *argp* depends on the selected request and is described for each request in detail later in this chapter.

The following ioctl codes are defined in *tpmc550.h*:

Value	Meaning
TPMC550_IOCSWRITE	Write output value
TPMC550_IOCGREADPARAM	Read the module configuration
TPMC550_IOCSSTARTSEQ	Start sequencer mode
TPMC550_IOCSWRITESEQ	Update sequencer output data
TPMC550_IOCSSTOPSEQ	Stop sequencer mode

See behind for more detailed information on each control code.

To use these TPMC550 specific control codes the header file tpmc550.h must be included in the application

RETURNS

On success, zero is returned. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

EINVAL	Invalid argument. This error code is returned if the requested ioctl function is unknown. Please check the argument *request*.

Other function dependant error codes will be described for each ioctl code separately. Note, the TPMC550 device driver always returns standard Linux error codes.

SEE ALSO

ioctl man pages

3.3.1 TPMC550_IOCSWRITE

NAME

TPMC550_IOCSWRITE – Write output value

DESCRIPTION

This ioctl function attempts to write the output value of the specified TPMC550 D/A channel.

A pointer to the caller's output buffer (*TPMC550_WRITEBUF*) is passed by the parameter *argp* to the driver.

```
typedef struct
{
        unsigned short          channel;
        unsigned short          flags;
        long                    value;
} TPMC550_WRITEBUF, *PTPMC550_WRITEBUF;
```

channel

This value specifies the DAC channel that will be used. Allowed values are 1 to 8 for TPMC551-10/-20 and 1 to 4 for TPMC551-11/-21.

flags

This value is an ORed value of the flags shown in the following table.

Name	Meaning
TPMC550_FL_CORR	If this flag is set, the driver will correct the DAC output value with the factory programmed correction data. If this flag is not set, the output value will not be corrected.
TPMC550_FL_LATCHED	It this flag is set the data will be loaded into the DAC, but the conversion will not be started, until the *TPMC550_FL_SIMCONV* flag is set.
TPMC550_FL_SIMCONV	This flag starts a simultaneous conversion for all channels. This flag is necessary to start a conversion in latched mode.

value

This parameter specifies the DAC output value (12bit LSB aligned). The value must be between 0 and 4095 for 0V..+10V mode and between –2048 and +2047 for –10V..+10V mode.

EXAMPLE

```
#include <tpmc550.h>

int                fd;
int                result;
TPMC550_WRITEBUF   DACBuf;

/*****************************************
 Write channel 5 with corrected data
*****************************************/
DACBuf.channel      = 5;
DACBuf.value        = 0x0123;
DACBuf.flags        = TPMC550_FL_CORR;
result = ioctl(fd, TPMC550_IOCSWRITE, &DACBuf);
if (result < 0)
{
     /* handle error */
     printf("\nFailed -->  Error = %d\n", errno);
}
```

ERRORS

EFAULT	Invalid pointer to the data buffer. Error copying data from user space.
ECHRNG	Invalid channel specified.
EBUSY	The sequencer mode is active on the specified device
ETIME	The settling or conversion time exceeds the supposed range.

SEE ALSO

ioctl man pages

3.3.2 TPMC550_IOCGREADPARAM

NAME

TPMC550_IOCGREADPARAM – Read the module configuration

DESCRIPTION

This ioctl function returns the module parameters. This includes the factory programmed correction data, number of channels and the voltage range selection.

A pointer to the callers parameter buffer (*TPMC550_PARABUF*) is passed by the parameter *argp* to the driver.

```
typedef struct
{
        int             NumChan;
        int             biPol_1_4;
        int             biPol_5_8;
        signed char     OffsCorr[8];
        signed char     GainCorr[8];
} TPMC550_PARABUF, *PTPMC550_PARABUF;
```

NumChan

This parameter returns the number of DAC channels supported by the module.

biPol_1_4

This parameter returns TRUE, if the channels 1 to 4 are configured for –10V..+10V mode, if FALSE is returned, the channels are configured for 0V..+10V mode.

biPol_5_8

This parameter returns TRUE, if the channels 5 to 8 are configured for –10V..+10V mode, if FALSE is returned, the channels are configured for 0V..+10V mode.

OffsCorr

This array returns the factory programmed offset correction data set, which is used if the *TPMC550_FL_CORR* flag is set. The index of the array specifies the channel number, 0 selects channel 1, 1 selects channel 2 and so on.

GainCorr

This array returns the factory programmed gain correction data set, which is used if the *TPMC550_FL_CORR* flag is set. The index of the array specifies the channel number, 0 selects channel 1, 1 selects channel 2 and so on.

EXAMPLE

```
#include <tpmc550.h>

int           fd;
int           result;
int           x;
TP551_PARABUF ParamBuf;

/*
** Read module configuration
*/
result = ioctl(fd, IOCGREADPARAM, &ParamBuf);
if (result >= 0)
{
     for (x = 0; ParamBuf.NumChan < 8; x++)
     {
          printf("Offset Error [%d] = %d \n", x + 1, ParamBuf.OffsCorr[x]);
          printf("Gain Error [%d] = %d \n", x + 1, ParamBuf.GainCorr[x]);
     }
} else {
     /* handle error */
     printf("\nFailed -->  Error = %d\n", errno);
}
```

ERRORS

EFAULT	Invalid pointer to the parameter buffer. Please check the argument argp.

SEE ALSO

ioctl man pages

3.3.3 TPMC550_IOCSSTARTSEQ

NAME

TPMC550_IOCSSTARTSEQ – Setup and start the sequencer, enter sequencer mode

DESCRIPTION

This ioctl function sets up the TPMC550 to work in sequencer mode. The cycle time and the channel configuration are set up.

A pointer to the callers parameter buffer (*TPMC550_STARTSEQBUF*) is passed by the parameter *argp* to the driver.

```
typedef struct
{
        unsigned short      channels;     /* channel number */
        unsigned short      cycleTime;    /* cycle time */
        unsigned short      flags;        /* flags */
} TPMC550_STARTSEQBUF, *PTPMC550_STARTSEQBUF;
```

channels

This parameter specifies which channel will be used in sequencer mode. Setting bit 0 will enable channel 1, setting bit 1 will enable channel 2 and so on.

cycleTime

This parameter specifies the cycle time that will be used. The value will be copied into the sequencer timer register. The value has a resolution of 100µs steps. If the flag TPMC550_FL_CONTINUOUS is set the parameter will be ignored (see below).

flags

This parameter is an ORed value of the following described flags.

Name	Meaning
TPMC550_FL_LATCHED	If this flag is set, the driver will output the data in latched mode, the output of all channels will be visible at the same time. Otherwise the data will be used in transparent mode.
TPMC550_FL_CONTINUOUS	The sequencer will work in continuous mode, data will be written as fast as possible to the output.

EXAMPLE

```
#include <tpmc550.h>

int  fd;
int  result;
TPMC550_STARTSEQBUF     SeqStartBuf;

/*************************************************************
 Start sequencer with a cycle time of 1 sec
 Enable following channels:
    Channel 1
    Channel 6
 Use latched mode
*************************************************************/
SeqStartBuf.cycleTime = 10000;    /* 10000 * 100µs */
SeqStartBuf.channels =  (1 << 0) | (1 << 5);    /* Enable channel */
SeqStartBuf.flags =     TPMC550_FL_LATCHED;
result = ioctl(fd, TPMC550_IOCSSTARTSEQ, &SeqStartBuf);
if (result < 0)
{
     /* handle error */
     printf("\nFailed -->  Error = %d\n", errno);
}
```

ERRORS

EFAULT	Invalid pointer to the parameter buffer. Please check the argument *argp*.

SEE ALSO

ioctl man pages

3.3.4 TPMC550_IOCSWRITESEQ

NAME

TPMC550_IOCSWRITESEQ – Write DAC data into sequencer FIFO-buffer

DESCRIPTION

This ioctl function writes data into the sequencer FIFO. The data will be used by the interrupt function in sequencer mode to update the DAC output values.

A pointer to the callers parameter buffer (*TPMC550_WRITESEQBUF*) is passed by the parameter *argp* to the driver.

```
typedef struct
{
    unsigned short      channels;     /* channel flags */
    unsigned short      correction;   /* correction flags */
    unsigned short      values[8];    /* buffer */
    unsigned long       stat;         /* write status */
} TPMC550_WRITESEQBUF, *PTPMC550_WRITESEQBUF;
```

channels

This parameter specifies which channel shall update output data. Setting bit 0 will update channel 1, setting bit 1 will update channel 2 and so on. Channels which are activated and not specified to be updated will hold their value.

correction

This parameter specifies which channels shall use the factory stored correction data. Setting bit 0 will enable data correction for channel 1, setting bit 1 will enable data correction for channel 2 and so on.

values

This array specifies the new output values. The array index specifies the channel number the data assigned to. Index 0 for channel 1, index 1 for channel 2 and so on. The values must be between 0 and 4095 for 0V..+10V mode and between –2048 and +2047 for –10V..+10V mode. Only the values for channels specified for update will be used.

stat

This parameter returns the sequencer status. The status returns number of cycles which had not been used for new data output, because there has been no output data available in the FIFO. And the status can signal that an output error has occurred. This will happen if the software is not able to handle a cycle before the next cycle starts. The stat argument is split in this way:

bits 27 .. 0	number of lost cycles
bit 30 (TPMC550_E_ERROR)	sequencer error has occurred

EXAMPLE

```
#include <tpmc550.h>

int  fd;
int  result;
TPMC550_WRITESEQBUF     SeqWriteBuf;

/*************************************************************
 Update Sequencer data
 Enable following channels:
    Channel 1
    Channel 6
 Use correction for channel 6
*************************************************************/
SeqWriteBuf.channels    = (1 << 0) | (1 << 5);
SeqWriteBuf.correction  = (1 << 5);
SeqWriteBuf.values[0]   = 0x0123;
SeqWriteBuf.values[5]   = 0x0000;
result = ioctl(fd, TPMC550_IOCSWRITESEQ, &SeqWriteBuf);
if (result < 0)
{
     /* handle error */
     printf("\nFailed -->  Error = %d\n", errno);
}
```

ERRORS

No function dependent errors.

SEE ALSO

ioctl man pages

3.3.5 TPMC550_IOCSSTOPSEQ

NAME

TPMC550_IOCSSTOPSEQ – Stop Sequencer Mode

DESCRIPTION

This ioctl function stops the sequencer mode.

The optional argument can be omitted for this ioctl function.

EXAMPLE

```
#include <tpmc550.h>

int  fd;
int  result;

/*
** stop sequencer mode
*/
result = ioctl(fd, TPMC550_IOCSSTOPSEQ);
if (result < 0)
{
     /* handle error */
     printf("\nFailed -->  Error = %d\n", errno);
}
```

ERRORS

No function dependent errors.

SEE ALSO

ioctl man pages

4 Diagnostic

If the TPMC550 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 displays information of a correct running TPMC550 device driver (see also the proc man pages).

```
# cat /proc/pci
PCI devices found:
...
  Bus  0, device  16, function  0:
    Signal processing controller: PLX Technology, Inc. PCI <-> IOBus Bridge
(rev 1).
      IRQ 25.
      Non-prefetchable 32 bit memory at 0xbffbff80 [0xbffbffff].
      I/O at 0xbfef00 [0xbfef7f].
      I/O at 0xbfeee0 [0xbfeeff].
      Non-prefetchable 32 bit memory at 0xbffbff60 [0xbffbff7f].

# cat /proc/devices
Character devices:
  1 mem
  2 pty/m%d
  3 pty/s%d
  4 tts/%d
  5 cua/%d
 10 misc
128 ptm
136 pts/%d
162 raw
254 tpmc550drv

# cat /proc/ioports
00000000-00bfffff : PCI host bridge
  00bfeee0-00bfeeff : PLX Technology, Inc. PCI <-> IOBus Bridge
    00bfeee0-00bfeeff : TPMC550
  00bfef00-00bfef7f : PLX Technology, Inc. PCI <-> IOBus Bridge
  00bfefc0-00bfefff : Intel Corp. 82559ER
    00bfefc0-00bfefff : eepro100
  00bff000-00bfffff : Tundra Semiconductor Corp. CA91C042 [Universe]
ffe80000-ffe80007 : serial(auto)
ffe80008-ffe8000f : serial(auto)
```

```
# cat /proc/iomem
80000000-ffffffff : PCI host bridge
  80000000-9fffffff : Universe VMEbus
    80000000-8000ffff : VME Master 0
    80010000-8020ffff : VME Master 1
  bffbff60-bffbff7f : PLX Technology, Inc. PCI <-> IOBus Bridge
  bffbff80-bffbffff : PLX Technology, Inc. PCI <-> IOBus Bridge
  bffc0000-bffdffff : Intel Corp. 82559ER
  bfffe000-bfffefff : Intel Corp. 82559ER
    bfffe000-bfffefff : eepro100
  bffff000-bfffffff : Tundra Semiconductor Corp. CA91C042 [Universe]
```