The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC700-SW-82

Linux Device Driver

32(16) Digital Output PMC

Version 1.0.x

User Manual

Issue 1.0.1

May 2009

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7
25469 Halstenbek, Germany
www.tews.com
Phone: +49 (0) 4101 4058 0
Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com

TEWS TECHNOLOGIES LLC
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com
Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TPMC700-SW-82

Linux Device Driver

32(16) Digital Output PMC

Supported Modules:
TPMC700

Issue	Description	Date
1.0.0	First Issue	December 1, 2006
1.0.1	General Revision	May 22, 2009

Table of Contents

1 Introduction

The TPMC700-SW-82 Linux device driver allows the operation of the TPMC700 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 TPMC700-SW-82 device driver supports the following features:

- writing a new output value
- enable and disable the output watchdog
- acknowledge watchdog errors

The TPMC700-SW-82 device driver supports the modules listed below:

TPMC700	32/16 Digital Outputs (24V, 0.5A) High Side Switches	(PMC)

In this document all supported modules and devices will be called TPMC700. Specials for a certain devices will be advised.

To get more information about the features and use of TPMC700 devices it is recommended to read the manuals listed below.

TPMC700 User manual

TPMC700 Engineering Manual

2 Installation

The directory TPMC700-SW-82 on the distribution media contains the following files:

TPMC700-SW-82-1.0.1.pdf	This manual in PDF format
TPMC700-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
Release.txt	Release information
ChangeLog.txt	Release history

The GZIP compressed archive TPMC700-SW-82-SRC.tar.gz contains the following files and directories:

Directory path 'tpmc700':

tpmc700.c	Driver source code
tpmc700def.h	Driver private include file
tpmc700.h	Driver public include file for application program
Makefile	Device driver make file
makenode	Script to create device nodes on the file system
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
include/config.h	Driver independent library header file
example/tpmc700exa.c	Example application
example/Makefile	Example application make file

In order to perform an installation, extract all files of the archive TPMC700-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC700-SW-82-SRC.tar.gz' will extract the files into the local directory. Copy the file 'tpmc700.h' into the directory */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 tpmc700drv

- 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 TPMC700 module found. The first TPMC700 module can be accessed with device node /dev/tpmc700_0, the second with device node /dev/tpmc700_1 and so on.

The assignment of device nodes to physical TPMC700 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 tpmc700drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc700_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 "*tpmc700drv: 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 TPMC700 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 tpmc700def.h, change the following symbol to appropriate value and enter `make install` to create a new driver.

TPMC700_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC700_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/tpmc700_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 <tpmc700.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 *tpmc700.h*:

Value	Meaning
TPMC700_IOC_WRITE	Write output value
TPMC700_IOC_WDENABLE	Enable output watchdog
TPMC700_IOC_WDDISABLE	Disable output watchdog
TPMC700_IOC_WDRESET	Reset output watchdog

See behind for more detailed information on each control code.

To use these TPMC700 specific control codes the header file tpmc700.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 TPMC700 device driver always returns standard Linux error codes.

SEE ALSO

ioctl man pages

3.3.1 TPMC700_IOC_WRITE

NAME

TPMC700_IOC_WRITE – Write output value

DESCRIPTION

This ioctl function attempts to write the output port of the TPMC700 associated with the open device handle.

A pointer to the caller's output value (*unsigned long*) is passed by the parameter *argp* to the driver. Bit 0 of the *unsigned long* value corresponds to TPMC700 output line 1, bit 1 corresponds to output line 2 and so on.

EXAMPLE

```
#include <tpmc700.h>

int               fd;
int               result;
unsigned long     OutputValue;

/*
** Write 0x12345678 to the output port.
*/
OutputValue = 0x12345678;
result = ioctl(fd, TPMC700_IOC_WRITE, &OutputValue);
if(result < 0)
{
     /* handle error */
}
```

ERRORS

EFAULT	Invalid pointer to the data buffer. Error copying data from user space.
EIO	The output register is locked by a watchdog failure. Execute the control function TPMC700_IOC_WDRESET to reset the watchdog error.

3.3.2 TPMC700_IOC_WDENABLE

NAME

TPMC700_IOC_WDENABLE – Enable output watchdog

DESCRIPTION

This ioctl function enables the output watchdog function of the TPMC700 after the next write operation to the device. Please remember if the watchdog is enabled and no write access occurs within 120 ms all outputs will be switched into the OFF state. To unlock the output register and leave the OFF state the device control function TPMC700_IOC_WDRESET must be executed.

The optional argument can be omitted for this ioctl function.

EXAMPLE

```
#include <tpmc700.h>

int  fd;
int  result;


/*
** Enable output watchdog.
*/
result = ioctl(fd, TPMC700_IOC_WDENABLE);
if(result < 0)
{
     /* handle error */
}
```

3.3.3 TPMC700_IOC_WDDISABLE

NAME

TPMC700_IOC_WDDISABLE – Disable output watchdog

DESCRIPTION

This ioctl function disables the output watchdog function of the TPMC700.

The optional argument can be omitted for this ioctl function.

EXAMPLE

```
#include <tpmc700.h>

int  fd;
int  result;


/*
** Disable output watchdog.
*/
result = ioctl(fd, TPMC700_IOC_WDDISABLE);
if(result < 0)
{
     /* handle error */
}
```

3.3.4 TPMC700_IOC_WDRESET

NAME

TPMC700_IOC_WDRESET – Reset output watchdog

DESCRIPTION

This ioctl function resets the output watchdog function of the TPMC700, and enables the outputs again.

The optional argument can be omitted for this ioctl function.

EXAMPLE

```
#include <tpmc700.h>

int  fd;
int  result;


/*
** Disable output watchdog.
*/
result = ioctl(fd, TPMC700_IOC_WDRESET);
if(result < 0)
{
     /* handle error */
}
```

4 Diagnostic

If the TPMC700 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 TPMC700 device driver (see also the proc man pages).

```
# cat /proc/pci
PCI devices found:
...
  Bus  0, device  16, function  0:
    Signal processing controller: PCI device 1498:02bc (TEWS Datentechnik
GmBH) (rev 1).
      IRQ 25.
      Non-prefetchable 32 bit memory at 0xbffbff80 [0xbffbffff].
      I/O at 0xbfef00 [0xbfef7f].
      Non-prefetchable 32 bit memory at 0xbffbff70 [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 tpmc700drv
```

```
# cat /proc/ioports
00000000-00bfffff : PCI host bridge
  00bfef00-00bfef7f : PCI device 1498:02bc (TEWS Datentechnik GmBH)
  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
  bffbff70-bffbff7f : PCI device 1498:02bc (TEWS Datentechnik GmBH)
    bffbff70-bffbff7f : TPMC700
  bffbff80-bffbffff : PCI device 1498:02bc (TEWS Datentechnik GmBH)
  bffc0000-bffdffff : Intel Corp. 82559ER
  bfffe000-bfffefff : Intel Corp. 82559ER
    bfffe000-bfffefff : eepro100
  bffff000-bfffffff : Tundra Semiconductor Corp. CA91C042 [Universe]
```