The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC700-SW-72

LynxOS Device Driver

32/16 Digital Outputs (24V, 0.5A)

High Side Switches

Version 1.0.x

User Manual

Issue 1.0.0

August 2009

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

TPMC700-SW-72

LynxOS Device Driver

32/16 Digital Outputs (24V, 0.5A)

Supported Modules:
TPMC700

Issue	Description	Date
1.0.0	First Issue	August 20, 2009

Table of Contents

1 INTRODUCTION 4
2 INSTALLATION 5
2.1 Device Driver Installation 6
2.1.1 Static Installation 6
2.1.1.1 Build the driver object 6
2.1.1.2 Create Device Information Declaration 6
2.1.1.3 Modify the Device and Driver Configuration File 6
2.1.1.4 Rebuild the Kernel 7
2.1.2 Dynamic Installation 7
2.1.2.1 Build the driver object 7
2.1.2.2 Create Device Information Declaration 7
2.1.2.3 Uninstall dynamic loaded driver 8
2.1.3 Device Information Definition File 8
2.1.4 Configuration File: CONFIG.TBL 9
3 TPMC700 DEVICE DRIVER PROGRAMMING 10
3.1 open() 10
3.2 close() 12
3.3 ioctl() 13
3.3.1 TPMC700_WRITE 14
3.3.2 TPMC700_LINESET 15
3.3.3 TPMC700_WDENABLE 17
3.3.4 TPMC700_WDDISABLE 18
3.3.5 TPMC700_WDRESET 19
4 DEBUGGING AND DIAGNOSTIC 20

1 Introduction

The TPMC700-SW-72 LynxOS device driver allows the operation of the TPMC700 Digital Output PMC conforming to the LynxOS I/O system specification. This includes a device-independent basic I/O interface with *open()*, *close()* and *ioctl()* functions.

The standard file (I/O) functions (*open()*, *close()*, *ioctl()*) provide the basic interface for opening and closing a file descriptor and for performing device I/O and configuration operations.

The TPMC700-SW-72 device driver supports the following features:

- write a new output state effecting all lines
- modify output state of selected lines (masked write)
- enabling, disabling output watchdog
- resetting watchdog state

The TPMC700-SW-72 device driver supports the modules listed below:

TPMC700-10	32 digital outputs (Front Panel I/O)	(PMC)
TPMC700-11	16 digital outputs (Front Panel I/O)	(PMC)
TPMC700-20	32 digital outputs (P14 I/O)	(PMC)
TPMC700-21	16 digital outputs (P14 I/O)	(PMC)

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

Following files are located on the distribution media:

Directory path 'TPMC700-SW-72':

TPMC700-SW-72-SRC.tar.gz	GZIP compressed archive with driver source code
TPMC700-SW-72-1.0.0.pdf	PDF copy of this manual
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 TPMC700-SW-72-SRC.tar.gz contains the following files and directories:

Directory path 'tpmc700':

tpmc700.c	TPMC700 device driver source
tpmc700def.h	TPMC700 driver include file
tpmc700.h	TPMC700 include file for driver and application
tpmc700_info.c	TPMC700 Device information definition
tpmc700_info.h	TPMC700 Device information definition header
tpmc700.cfg	TPMC700 Driver configuration file include
tpmc700.import	Linker import file
Makefile	Device driver make file
example/tpmc700exa.c	Example application
example/Makefile	Example application makefile

In order to perform a driver installation, first extract the TAR file to a temporary directory, than follow the steps below:

1. Create a new directory in the system drivers directory path /sys/drivers.xxx, where xxx represents the BSP that supports the target hardware.

   For example: /sys/drivers.pp_drm/tpmc700 or /sys/drivers.cpci_x86/tpmc700

2. Copy the following files to this directory:
   - tpmc700.c
   - tpmc700def.h
   - tpmc700.import
   - Makefile

3. Copy tpmc700.h to /usr/include/

4. Copy tpmc700_info.c to /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).

5. Copy tpmc700_info.h to /sys/dheaders/

Copy tpmc700.cfg to */sys/cfg.xxx/*, where xxx represents the BSP for the target platform. For example: /sys/cfg.ppc or /sys/cfg.x86 ....

2.1 Device Driver Installation

The two methods of driver installation are as follows:

(1) Static Installation
(2) Dynamic Installation (only native LynxOS 4 systems)

2.1.1 Static Installation

With this method, the driver object code is linked with the kernel routines and is installed during system start-up.

2.1.1.1 Build the driver object

1. Change to the directory /sys/drivers.xxx/tpmc700, where xxx represents the BSP that supports the target hardware.

2. To update the library /sys/lib/libdrivers.a enter:

```
make install
```

2.1.1.2 Create Device Information Declaration

1. Change to the directory /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).

2. Add the following dependencies to the Makefile

```
DEVICE_FILES_all = … tpmc700_info.x
```

   And at the end of the Makefile

```
tpmc700_info.o:$(DHEADERS)/tpmc700_info.h
```

3. To update the library /sys/lib/libdevices.a enter:

```
make install
```

2.1.1.3 Modify the Device and Driver Configuration File

In order to insert the driver object code into the kernel image, an appropriate entry in file CONFIG.TBL must be created.

1. Change to the directory /sys/lynx.os/ respective /sys/bsp.xxx, where xxx represents the BSP that supports the target hardware.

2. Create an entry at the end of the file CONFIG.TBL

   Insert the following entry at the end of this file.

```
I:tpmc700.cfg
```

2.1.1.4 Rebuild the Kernel

1. Change to the directory /sys/lynx.os/ (/sys/bsp.xxx)
2. Enter the following command to rebuild the kernel:

   ```
   make install
   ```

3. Reboot the newly created operating system by the following command (not necessary for KDIs):

   ```
   reboot –aN
   ```

   The N flag instructs init to run mknod and create all the nodes mentioned in the new nodetab.

4. After reboot you should find the following new devices (depends on the device configuration): /dev/tpmc700a, /dev/tpmc700b, …

2.1.2 Dynamic Installation

This method allows you to install the driver after the operating system is booted. The driver object code is attached to the end of the kernel image and the operating system dynamically adds this driver to its internal structures. The driver can also be removed dynamically.

2.1.2.1 Build the driver object

1. Change to the directory /sys/drivers.xxx/tpmc700, where xxx represents the BSP that supports the target hardware.
2. To make the dynamic link-able driver enter:

   ```
   make dldd
   ```

2.1.2.2 Create Device Information Declaration

1. Change to the directory /sys/drivers.xxx/tpmc700, where xxx represents the BSP that supports the target hardware.
2. To create a device definition file for the major device (this works only on native systems)

   ```
   make t700info
   ```

3. To install the driver enter:

   ```
   drinstall –c tpmc700.obj
   ```

   If successful, drinstall returns a unique <driver-ID>

4. To install the major device enter:

   ```
   devinstall –c –d <driver-ID> t700info
   ```

   The <driver-ID> is returned by the drinstall command

5. To create the node for the devices enter:

   ```
   mknod /dev/tpmc700a c <major_no> 0
   ```

   The <major_no> is returned by the devinstall command.

If all steps are successfully completed, the TPMC700 is ready to use.

2.1.2.3 Uninstall dynamic loaded driver

To uninstall the TPMC700 device enter the following commands:

```
devinstall –u –c <device-ID>
drinstall –u <driver-ID>
```

2.1.3 Device Information Definition File

The device information definition contains information necessary to install the TPMC700 major device.

The implementation of the device information definition is done through a C structure, which is defined in the header file *tpmc700_info.h*.

This structure contains the following parameter:

PCIBusNumber	Contains the PCI bus number at which the supported device is connected. Valid bus numbers are in range from 0 to 255.
PCIDeviceNumber	Contains the device number (slot) at which the supported device is connected. Valid device numbers are in range from 0 to 31.

If both PCIBusNumber and PCIDeviceNumber are –1 then the driver will auto scan for supported devices. The first device found in the scan order will be allocated by the driver for this major device.

Already allocated devices can't be allocated twice. This is important to know if there are more than one TPMC700 major devices.

A device information definition is unique for every TPMC700 major device. The file *tpmc700_info.c* on the distribution media contains two device information declarations, **tpmc700A** for the first major device and **tpmc700B** for the second major device.

If the driver should support more than two major devices it is necessary to copy and paste an existing declaration and rename it with a unique name, for example **tpmc700C**, **tpmc700D** and so on.

It is also necessary to modify the device and driver configuration file, respectively the configuration include file *tpmc700.cfg*.

The following device declaration information uses the auto find method to detect a supported device on the PCI bus.

```
TPMC700_INFO tpmc700A = {

     -1,           /*  Auto find the device on any PCI bus   */
     -1,
};
```

2.1.4 Configuration File: CONFIG.TBL

The device and driver configuration file CONFIG.TBL contains entries for device drivers and its major and minor device declarations. Each time the system is rebuild, the config utility read this file and produces a new set of driver and device configuration tables and a corresponding nodetab.

To install the TPMC700 driver and devices into the LynxOS system, the configuration include file tpmc700.cfg must be included in the CONFIG.TBL (see also chapter 2.1.1.3).

The file tpmc700.cfg on the distribution disk contains the driver entry (*C:tpmc700:\....*) and two major device entries (*D:TPMC700 1:tpmc700A:: and D:TPMC700 2:tpmc700B::*).

If the driver should support more than one major device, the following entries for major devices must be enabled by removing the comment character (#). By copy and paste an existing major and minor entries and renaming the new entries, it is possible to add any number of additional TPMC700 devices.

This example shows a driver entry with two major devices and one minor device:

```
#    Format:
#    C:driver-name:open:close:read:write:select:control:install:uninstall
#    D:device-name:info-block-name:raw-partner-name
#    N:node-name:minor-dev

C:tpmc700:tpmc700open:tpmc700close: \
::: \
::tpmc700ioctl: \
:tpmc700install:tpmc700uninstall
D:TPMC700 1:tpmc700A::
N:tpmc700a:0
D:TPMC700 2:tpmc700B::
N:tpmc700b:0
```

The configuration above creates the following nodes in the /dev directory.

```
/dev/tpmc700a /dev/tpmc700b
```

3 TPMC700 Device Driver Programming

LynxOS system calls are all available directly to any C program. They are implemented as ordinary function calls to "glue" routines in the system library, which trap to the OS code.

Note that many system calls use data structures, which should be obtained in a program from appropriate header files. Necessary header files are listed with the system call synopsis.

3.1 open()

NAME

open() - open a file

SYNOPSIS

#include <sys/file.h>
#include <sys/types.h>
#include <fcntl.h>

int open (char *path, int oflags[, mode_t mode])

DESCRIPTION

Opens a file (TPMC700 device) named in ***path*** for reading and writing. The value of ***oflags*** indicates the intended use of the file. In case of a TPMC700 device ***oflags*** must be set to **O_RDWR** to open the file for both reading and writing.

The ***mode*** argument is required only when a file is created. Because a TPMC700 device already exists this argument is ignored.

EXAMPLE

```
int fd

fd = open ("/dev/tpmc700a", O_RDWR);
if (fd == -1)
{
    /* Handle error */
}
```

RETURNS

open() returns a file descriptor number if successful, or –1 on error.

SEE ALSO

LynxOS System Call - open()

3.2 close()

NAME

close() – close a file

SYNOPSIS

int close(int fd)

DESCRIPTION

This function closes an opened device.

EXAMPLE

```
int  result;

result = close(fd);
if (result == -1)
{
     /* Handle error */
}
```

RETURNS

close() returns 0 (OK) if successful, or –1 on error

SEE ALSO

LynxOS System Call - close()

3.3 ioctl()

NAME

ioctl() – I/O device control

SYNOPSIS

#include <ioctl.h>
#include <tpmc700.h>

int ioctl (int fd, int request, char *arg)

DESCRIPTION

ioctl provides a way of sending special commands to a device driver. The call sends the value of request and the pointer arg to the device associated with the descriptor fd.

The following ioctl codes are supported by the driver and are defined in *tpmc700.h*:

Symbol	**Meaning**
TPMC700_WRITE	Write new output value (all lines)
TPMC700_LINESET	Modify output lines (masked write)
TPMC700_WDENABLE	Enable output watchdog
TPMC700_WDDISABLE	Disable output watchdog
TPMC700_WDRESET	Reset output watchdog state

See behind for more detailed information on each control code.

RETURNS

ioctl returns 0 if successful, or –1 on error.

On error, *errno* will contain a standard error code (see also LynxOS System Call – ioctl).

SEE ALSO

LynxOS System Call - ioctl().

3.3.1 TPMC700_WRITE

NAME

TPMC700_WRITE – Write output value

DESCRIPTION

This function sets the state of all output lines. A pointer to the new output value (unsigned int) must be passed by the parameter *arg* to the device. Bit-0 of the value specifies OUT 1, bit-1 specifies OUT 2, and so on.

EXAMPLE

```
#include <tpmc700.h>

int                     fd;
int                     result;
unsigned int            outVal;

/* --- set the output value --- */
outVal = 0x12345678;
result = ioctl(fd, TPMC700_WRITE, (char*)&outVal);
if (result >= 0)
{
    /* Output lines are set */
}
else
{
    /* Setting output lines failed */
}
```

ERRORS

ETIMEDOUT	The output value can not be set. The watchdog has expired and does not allow a write before the watchdog state is reset.

Other returned error codes are system error conditions.

3.3.2 TPMC700_LINESET

NAME

TPMC700_LINESET – Modify output value (selected lines)

DESCRIPTION

This function modifies the state of specified output lines. A pointer to the callers write buffer (*TPMC700_LINESET_BUFFER*) must be passed by the parameter *arg* to the device.

```
typedef struct
{
        unsigned int        value;        /* new output value */
        unsigned int        mask;         /* mask for bits that shall be effected */
} TPMC700_LINESET_BUFFER, *PTPMC700_LINESET_BUFFER;
```

Members

value

This value specifies the new output value. Only the bits that are masked by *mask* will be used, the other bits are don't care. Bit-0 specifies OUT 1, bit-1 specifies OUT 2, and so on.

mask

This value specifies which lines shall be modified. A set bit means that the output line shall be set to the specified value, a reset bit means, that the line shall not be affected. Bit-0 specifies OUT 1, bit-1 specifies OUT 2, and so on.

EXAMPLE

```
#include <tpmc700.h>

int                     fd;
int                     result;
TPMC700_LINESET_BUFFER  wrBuf;

…
```

…

```
/* --- change the output value of OUT 1-8 --- */
/* ---     set odd lines                  --- */
/* ---     clear even lines               --- */
wrBuf.value = 0xAAAAAAAA;
wrBuf.mask  = 0x000000FF;
result = ioctl(fd, TPMC700_LINESET, (char*)&wrBuf);
if (result >= 0)
{
     /* Output lines set */
}
else
{
     /* Setting output lines failed */
}
```

ERRORS

ETIMEDOUT	The output value can not be set. The watchdog has expired and does not allow a lineset before the watchdog state is reset.

Other returned error codes are system error conditions.

3.3.3 TPMC700_WDENABLE

NAME

TPMC700_WDENABLE – Enable output watchdog

DESCRIPTION

This function enables the output watchdog. There are no parameters that must be passed by the parameter *arg* to the device.

The watchdog is triggered with every *TPMC700_WRITE* and *TPMC700_LINESET* command.

EXAMPLE

```
#include <tpmc700.h>

int             fd;
int             result;

/* --- enable watchdog --- */
result = ioctl(fd, TPMC700_WDENABLE, (char*)0);
if (result >= 0)
{
     /* Watchdog enabled */
}
else
{
     /* Enable watchdog failed */
}
```

3.3.4 TPMC700_WDDISABLE

NAME

TPMC700_WDDISABLE – Disable output watchdog

DESCRIPTION

This function disables the output watchdog. There are no parameters that must be passed by the parameter *arg* to the device.

EXAMPLE

```
#include <tpmc700.h>

int             fd;
int             result;

/* --- disable watchdog --- */
result = ioctl(fd, TPMC700_WDDISABLE, (char*)0);
if (result >= 0)
{
     /* Watchdog disabled */
}
else
{
     /* Disable watchdog failed */
}
```

3.3.5 TPMC700_WDRESET

NAME

TPMC700_WDRESET – Resets watchdog state

DESCRIPTION

This function resets the watchdog state. This function must be executed after a watchdog timeout has occurred, before a new value can be written to the board. There are no parameters that must be passed by the parameter *arg* to the device.

EXAMPLE

```
#include <tpmc700.h>

int              fd;
int              result;

/* --- reset watchdog state --- */
result = ioctl(fd, TPMC700_WDRESET, (char*)0);
if (result >= 0)
{
     /* Watchdog state resetted */
}
else
{
     /* Reset watchdog state failed */
}
```

4 Debugging and Diagnostic

If the driver will not work properly, please enable debug outputs by defining the symbols *DEBUG, DEBUG_TPMC,* and *DEBUG_PCI* in file tpmc700.c.

The debug output should appear on the console. If not, please check the symbol *KKPF_PORT* in *uparam.h*. This symbol should be configured to a valid COM port (e.g. *SKDB_COM1*).

The debug output displays the device information data for the current major device like this.

```
TPMC700: Device Driver Install
Bus = 1  Dev = 2  Func = 0
[00] = 02BC1498
[04] = 02800000
[08] = 11800001
[0C] = 00000000
[10] = 84000000
[14] = 00804001
[18] = 84001000
[1C] = 00000000
[20] = 00000000
[24] = 00000000
[28] = 00000000
[2C] = 000A1498
[30] = 00000000
[34] = 00000000
[38] = 00000000
[3C] = 000000FF

PCI Base Address 0 (PCI_RESID_BAR0)
B4600000 : F0 FF FF 0F 00 00 00 00 00 00 00 00 00 00 00 00
B4600010 : 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
B4600020 : 00 00 00 00 00 00 00 00 80 78 B1 01 00 00 00 00
B4600030 : 00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00

PCI Base Address 1 (PCI_RESID_BAR1)

PCI Base Address 2 (PCI_RESID_BAR2)
B4601000 : 00000000 00000000

TPMC700: Found TPMC700-10 on Bus 1, Device 2
```

The debug output above is only an example. Debug output on other systems may be different for addresses and data in some locations.