The Embedded I/O Company

TPMC685-SW-82

Linux Device Driver

16 x 8 Bit Digital Inputs/Outputs (5V TTL)

Version 1.0.x

User Manual

Issue 1.0.0

August 2020

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

TPMC685-SW-82

Linux Device Driver

16 x 8 Bit Digital Inputs/Outputs (5V TTL)

Supported Modules:
TPMC685

Issue	Description	Date
1.0.0	First Issue	August 28, 2020

Table of Contents

1 Introduction

The TPMC685-SW-82 Linux device driver provides support for the TEWS TECHNOLOGIES PMC modules with 128 digital I/O lines listed below.

In addition to the low-level device driver, an Application Programming Interface (API) is provided for system independent interface.

The purpose of this document is to describe the device driver functionality, especially the Application Programming Interface.

The TPMC685-SW-82 Linux device driver and its API support the following features:

- read input value from port
- write output value to port
- read input values from selected ports
- write input values to selected ports
- set and reset output value of a single I/O line
- update input and output of simultaneous ports
- configure port direction
- configure 64-bit simultaneous input and output
- set pullup/pulldown configuration
- configure input filtering
- configure output watchdog function
- wait for input events
- wait for watchdog events
- read timer value
- configure timers
- wait for timer events

The TPMC685-SW-82 device driver supports the modules listed below:

TPMC685	16 x 8 Bit Digital Inputs/Outputs	PMC

To get more information about the features and use of TPMC685 device it is recommended to read the manuals listed below.

TPMC685 User Manual

2 Installation

The directory TPMC685-SW-82 on the distribution media contains the following files:

TPMC685-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
TPMC685-SW-82-1.0.0.pdf	PDF copy of this manual
Release.txt	Release information
ChangeLog.txt	Release history

The GZIP compressed archive TPMC685-SW-82-SRC.tar.gz contains the following files and directories:

Directory path ‘tpmc685’:

tpmc685.c	TPMC685 device driver source
tpmc685def.h	TPMC685 driver include file
tpmc685.h	TPMC685 include file for driver and application
Makefile	Device driver make file
makenode	Script to create device nodes on the file system
COPYING	Copy of the GNU Public License (GPL)
api/tpmc685api.h	API include file
api/tpmc685api.c	API source file
include/config.h	Driver independent library 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
example/tpmc685exa.c	Example application
example/Makefile	Example application make file

In order to perform an installation, extract all files of the archive TPMC685-SW-82-SRC.tar.gz to the desired target directory. The command ‘tar -xzvf TPMC685-SW-82-SRC.tar.gz’ will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tpmc685.h and api/tpmc685api.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 tpmc685drv

- 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 dynamic 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 TPMC685 device found. The first TPMC685 module can be accessed with device node /dev/tpmc685_0, the second module with device node /dev/tpmc685_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 tpmc685drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc685_* 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 "*tpmc685drv: 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 TPMC685 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 *tpmc685def.h*, change the following symbol to appropriate value and enter **make install** to create a new driver.

TPMC685_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC685_MAJOR   122
```

Be sure that the desired major number isn't used by other drivers. Please check /proc/devices to see which numbers are free.

3 API Documentation

3.1 General Functions

3.1.1 tpmc685Open

NAME

tpmc685Open – open a device.

SYNOPSIS

```
TPMC685_HANDLE tpmc685Open
(
        char            *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle must be opened by a call to this function.

The tpmc685Open function can be called multiple times (e.g. in different tasks).

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TPMC685 device is named “/dev/tpmc685_0”, the second device is named “/dev/tpmc685_1” and so on.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tpmc685Open(“/dev/tpmc685_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 tpmc685Close

NAME

tpmc685Close – close a device.

SYNOPSIS

```
TPMC685_STATUS tpmc685Close
(
    TPMC685_HANDLE      hdl
)
```

DESCRIPTION

This function closes a previously opened device.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/*
** close the device
*/
result = tpmc685Close(hdl);
if (result != TPMC685_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 tpmc685GetPciInfo

NAME

tpmc685GetPciInfo – get information about the module's PCI configuration

SYNOPSIS

```
TPMC685_STATUS tpmc685GetPciInfo
(
        TPMC685_HANDLE              hdl,
        TPMC685_PCIINFO_BUF         *pPciInfoBuf
)
```

DESCRIPTION

This function returns information about the module's PCI configuration in the provided data buffer.

PARAMETERS

hdl

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

pPciInfoBuf

This argument is a pointer to the structure TPMC685_PCIINFO_BUF that receives information about the module PCI header.

```
typedef struct
{
        unsigned short      vendorId;
        unsigned short      deviceId;
        unsigned short      subSystemId;
        unsigned short      subSystemVendorId;
        int                 pciBusNo;
        int                 pciDevNo;
        int                 pciFuncNo;
} TPMC685_PCIINFO_BUF;
```

vendorId

PCI module vendor ID

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module sub system vendor ID

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE          hdl;
TPMC685_STATUS          result;
TPMC685_PCIINFO_BUF     pciInfoBuf

/*
** get module PCI information
*/
result = tpmc685GetPciInfo(hdl, &pciInfoBuf);
if (result == TPMC685_OK)
{
     printf(“PCI-Location: Bus: %d – Dev: %d – Func: %d\n”,
          pciInfoBuf.pciBusNo, pciInfoBuf.pciDevNo, pciInfoBuf.pciFuncNo);
     …
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.1.4 tpmc685GetBoardInfo

NAME

tpmc685GetBoardInfo – get information about the module

SYNOPSIS

```
TPMC685_STATUS tpmc685GetBoardInfo
(
        TPMC685_HANDLE              hdl,
        unsigned short              *fpgaVersion
)
```

DESCRIPTION

This function returns information about the module.

PARAMETERS

hdl

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

fpgaVersion

This argument is a pointer to an *unsigned short* value, which will be filled with the module's FPGA-Code Version (CODE_VER of the Global Control Register).

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE         hdl;
TPMC685_STATUS         result;
unsigned short         fpgaVersion

…
```

```
…

/*
** get module PCI information
*/
result = tpmc685GetBoardInfo(hdl, &fpgaVersion);
if (result == TPMC685_OK)
{
     printf("FPGA-Code Version: %04X\n", fpgaVersion);
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.2 Digital I/O Configuration Functions

3.2.1 tpmc685ConfigPortDir

NAME

tpmc685ConfigPortDir – configure the direction of ports

SYNOPSIS

```
TPMC685_STATUS tpmc685ConfigPortDir
(
        TPMC685_HANDLE              hdl,
        unsigned short              direction,
        unsigned short              mask
)
```

DESCRIPTION

Configure the direction of the I/O ports. Affected ports can be selected with a mask.

PARAMETERS

hdl

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

direction

This value specifies the direction of the ports. A set bit means that port shall be configured for output and a reset bit will configure the port for input. Bit-0 represents port 0, bit-1 represents port 1, and so on.

mask

This value masks the affected ports. A set bit means that port shall be affected and a reset bit will keep the port direction unchanged. Bit-0 represents port 0, bit-1 represents port 1, and so on.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned short     direction;
unsigned short     mask;

/* Configure port 3,4 for output and 6,7 for input */
direction = (1 << 3) | (1 << 4) | (0 << 6) | (0 << 7);
mask      = (1 << 3) | (1 << 4) | (1 << 6) | (1 << 7);
result = tpmc685ConfigPortDir (hdl, direction, mask);
if (result != TPMC685_OK)
{
    /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.2.2 tpmc685ConfigPortPull

NAME

tpmc685ConfigPortPull – configure port I/O pull configuration

SYNOPSIS

```
TPMC685_STATUS tpmc685ConfigPortPull
(
        TPMC685_HANDLE          hdl,
        TPMC685_PULLMODE        portPullMode[16]
)
```

DESCRIPTION

This function configures if the I/O lines of a port are pulled down, or pulled up to 3.3V or 5V, or if they are not pulled.

PARAMETERS

hdl

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

portPullMode

The parameter points to an array of TPMC685_PULLMODE values. The values in the array define the pull-up/-down mode of the ports. Element index 0 assigns the mode of port 0, index 1 assigns the mode of port 1, and so on. The following pull modes can be assigned:

Pull Mode	Description
TPMC685_PULL_OFF	The I/O lines of the port are neither pulled down nor pulled up.
TPMC685_PULL_DOWN	The I/O lines of the port are pulled down.
TPMC685_PULL_UP_3V	The I/O lines of the port are pulled up to 3.3V.
TPMC685_PULL_UP_5V	The I/O lines of the port are pulled up to 5V.

EXAMPLE

```
#include "tpmc685api.h"

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
TPMC685_PULLMODE   portPullModeArray[16] = {
                              TPMC685_PULL_OFF,   TPMC685_PULL_OFF,
                              TPMC685_PULL_OFF,   TPMC685_PULL_OFF,
                              TPMC685_PULL_OFF,   TPMC685_PULL_DOWN,
                              TPMC685_PULL_UP_3V, TPMC685_PULL_UP_3V,
                              TPMC685_PULL_UP_5V, TPMC685_PULL_UP_5V,
                              TPMC685_PULL_UP_5V, TPMC685_PULL_UP_5V,
                              TPMC685_PULL_UP_5V, TPMC685_PULL_UP_5V ,
                              TPMC685_PULL_UP_5V, TPMC685_PULL_UP_5V };

/*
** configure the Port Pull Modes
**    - Port 0...4  – no pull
**    - Port 5      – pull down
**    - Port 6...7  – pull up (3.3V)
**    - Port 8...15 – pull up (5V)
*/
result = tpmc685ConfigPortPull(hdl, portPullModeArray);
if (result != TPMC685_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.2.3 tpmc685ConfigSimPorts

NAME

tpmc685ConfigSimPorts – configure the simultaneous mode of ports

SYNOPSIS

```
TPMC685_STATUS tpmc685ConfigSimPorts
(
    TPMC685_HANDLE          hdl,
    unsigned int            simMode
)
```

DESCRIPTION

Enables or disables simultaneous mode for 64-bit port groups.

PARAMETERS

hdl

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

simMode

This value specifies the simultaneous behavior of the board. The following definitions set the new behavior for 64-I/O lines (groups of 8 ports) and can be ORed to configure input and output of both groups within one call.

Definition	**Description**
TPMC685_SIMMOD_FRONTIN_ENABLE	Enable simultaneous input mode for port 0 … port 7 (Front I/O ports)
TPMC685_SIMMOD_FRONTIN_DISABLE	Disable simultaneous input mode for port 0 … port 7 (Front I/O ports)
TPMC685_SIMMOD_REARIN_ENABLE	Enable simultaneous input mode for port 8 … port 15 (Rear I/O ports)
TPMC685_SIMMOD_REARIN_DISABLE	Disable simultaneous input mode for port 8 … port 15 (Rear I/O ports)
… continued	

TPMC685_SIMMOD_FRONTOUT_ENABLE	Enable simultaneous output mode for port 0 … port 7 (Front I/O ports)
TPMC685_SIMMOD_FRONTOUT_DISABLE	Disable simultaneous output mode for port 0 … port 7 (Front I/O ports)
TPMC685_SIMMOD_REAROUT_ENABLE	Enable simultaneous output mode for port 8 … port 15 (Rear I/O ports)
TPMC685_SIMMOD_REAROUT_DISABLE	Disable simultaneous output mode for port 8 … port 15 (Rear I/O ports)

Note: Do not use “ENABLE” and “DISABLE” definitions for the same port group and direction at once.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/*
** Configure front I/O port for simultaneous input and
** disable simultaneous output for Rear I/O ports ---
** front output and rear input mode shall not be modified
*/
result = tpmc685ConfigSimPort (hdl, TPMC685_SIMMOD_FRONTIN_ENABLE |
                                    TPMC685_SIMMOD_REAROUT_DISABLE);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.2.4 tpmc685ConfigPortFilter

NAME

tpmc685ConfigPortFilter – configure an input port filter

SYNOPSIS

```
TPMC685_STATUS tpmc685ConfigPortFilter
(
        TPMC685_HANDLE          hdl,
        unsigned int            portNo,
        TPMC685_ONOFF           filterEnable,
        TPMC685_TIMEBASE        filterBase,
        unsigned short          filterTime
)
```

DESCRIPTION

This function configures the input filter for the specified port.

PARAMETERS

hdl

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

portNo

This value selects the port number that shall be used. Valid port numbers are 0 up to 15.

filterEnable

This value specifies if the input filter is active or not. The values are defined:

Mode	**Description**
TPMC685_ENABLE	The input filter will be enabled for the specified port.
TPMC685_DISABLE	The input filter will be disabled for the specified port, all following parameters a not used

filterBase

This value specifies the time base of the filter delay. Possible values are:

Filter Time Base	**Description**
TPMC685_1NANOSEC	Timer base is 1ns (lowest 7 bits of the filterTime parameter will be ignored) (steps of ~120ns)
TPMC685_1MICROSEC	Timer base is 1µs
TPMC685_1MILLISEC	Timer base is 1ms

filterTime

This value specifies the filter time value based on selected timer base in *filterBase*. The filter time value has a width of 16-bit.

EXAMPLE

```
#include "tpmc685api.h"

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/*
** filter input of port 3 for 5µsec
*/
result = tpmc685ConfigPortFilter(hdl,
                                 3
                                 TPMC685_ENABLE,
                                 TPMC685_1MICROSEC,
                                 5);
if (result != TPMC685_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3 Digital I/O Functions

3.3.1 tpmc685ReadPort

NAME

tpmc685ReadPort – read the input value of a port

SYNOPSIS

```
TPMC685_STATUS tpmc685ReadPort
(
        TPMC685_HANDLE          hdl,
        unsigned int            portNo,
        unsigned char           *data
)
```

DESCRIPTION

Read the current state of the digital lines of a specified port.

If the port is part of simultaneous read ports (64-bit), this function will return the latched port value.

PARAMETERS

hdl

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

portNo

This value selects the port number that shall be used. Valid port numbers are 0 up to 15.

data

This parameter points to an *unsigned char* value where the port's data register content is stored. Bit-0 represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned char      inputValue;

/* read the current input value of port 4 */
result = tpmc685ReadPort(hdl, 4, &inputValue);
if (result == TPMC685_OK)
{
     printf(“Port 4 - Input Value = 0x%02X\n”, inputValue);
}
else
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3.2 tpmc685WritePort

NAME

tpmc685WritePort – write the output value of a port

SYNOPSIS

```
TPMC685_STATUS tpmc685WritePort
(
        TPMC685_HANDLE              hdl,
        unsigned int                portNo,
        unsigned char               data
)
```

DESCRIPTION

Write a new output value to the lines of a specified port. The function returns immediately to the caller.

If the port is part of simultaneous write port (64-bit), this function will just renews the latched output value, but not update the output state of the I/O lines.

There will be no effect if the port is not configured as output port.

PARAMETERS

hdl

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

portNo

This value selects the port number that shall be used. Valid port numbers are 0 up to 15.

data

This value specifies the new output value. Bit-0 represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned char      outputValue;

/* set value of port 3 (set I/O line 1 and 2) */
outputValue = (1 << 1) | (1 << 2);
result = tpmc685WritePort(hdl, 3, outputValue);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3.3 tpmc685ReadMaskedPorts

NAME

tpmc685ReadMaskedPorts – read the input values of selected ports

SYNOPSIS

```
TPMC685_STATUS tpmc685ReadMaskedPorts
(
      TPMC685_HANDLE            hdl,
      unsigned short            portMask,
      unsigned char             data[TPMC685_NUM_PORTS]
)
```

DESCRIPTION

Read the current state of the digital lines of selected I/O ports.

If selected ports are part of simultaneous read ports (64-bit), this function will return for these ports the latched port value.

PARAMETERS

hdl

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

portMask

This value specifies ports that shall be read. A set bit specifies that the corresponding port shall be used, a reset bit announces to skip the port. Bit-0 corresponds to port 0, bit-1 to port 1, and so on.

data

This parameter points to an array of *unsigned char* values with sixteen elements where the port's data register content is stored. Element index 0 represents port 0, index 1 represents port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned short     portMask;
unsigned char      inputValues[TPMC685_NUM_PORTS];

/* read the current input values of port 0...7 and 12 */
portMask = (0xFF << 0) | (1 << 12);
result = tpmc685ReadMaskedPorts(hdl, portMask, inputValues);
if (result == TPMC685_OK)
{
     printf(“Port  0 - Input Value = 0x%02X\n”, inputValue[0]);
     printf(“Port  1 - Input Value = 0x%02X\n”, inputValue[1]);
     ...
     printf(“Port 12 - Input Value = 0x%02X\n”, inputValue[12]);
}
else
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3.4 tpmc685WriteMaskedPorts

NAME

tpmc685WriteMaskedPorts – write the output values of selected ports

SYNOPSIS

```
TPMC685_STATUS tpmc685WriteMaskedPorts
(
    TPMC685_HANDLE          hdl,
    unsigned short          portMask,
    unsigned char           data[TPMC685_NUM_PORTS]
)
```

DESCRIPTION

Write the values of the digital lines on to the selected I/O ports.

If selected ports are part of simultaneous write ports (64-bit), this function will just renew the latched output value of these ports, but not update the output state of the I/O lines.

Ports not configured as output will not be effected.

PARAMETERS

hdl

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

portMask

This value specifies ports that shall be written. A set bit specifies that the corresponding port shall be used, a reset bit announces to skip the port. Bit-0 corresponds to port 0, bit-1 to port 1, and so on.

data

This parameter contains an array of *unsigned char* values with sixteen elements where the port's new output data is specified. Element index 0 represents port 0, index 1 represents port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE      hdl;
TPMC685_STATUS      result;
unsigned short      portMask;
unsigned char       outputValues[TPMC685_NUM_PORTS] =
                              {0,1,2,3,4,5,6,7,8,9,0xA,0xB,0xC,0xD,0xE,0xF};

/* write output values for port 0...7 and 11 */
portMask = (0xFF << 0) | (1 << 11);
result = tpmc685WriteMaskedPorts(hdl, portMask , outputValues);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3.5 tpmc685SetLine

NAME

tpmc685SetLine – Set a single I/O line

SYNOPSIS

```
TPMC685_STATUS tpmc685SetLine
(
        TPMC685_HANDLE          hdl,
        unsigned int            portNo,
        unsigned int            lineNo
)
```

DESCRIPTION

Set a specified I/O line.

If the selected port is part of simultaneous write ports (64-bit), this function will just modify the latched output value, but not update the output state of the I/O line.

There will be no effect if the I/O line is not configured as output port.

PARAMETERS

hdl

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

portNo

This value selects the port number that shall be used. Valid port numbers are 0 up to 15.

lineNo

This value selects the I/O line that shall be used. Valid line numbers are 0 up to 7.

EXAMPLE

```
#include "tpmc685api.h"

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/* set I/O-line 4 of port 12 */
result = tpmc685SetLine(hdl, 12, 4);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3.6 tpmc685ResetLine

NAME

tpmc685SetLine – Reset a single I/O line

SYNOPSIS

```
TPMC685_STATUS tpmc685ResetLine
(
        TPMC685_HANDLE          hdl,
        unsigned int            portNo,
        unsigned int            lineNo
)
```

DESCRIPTION

Reset a specified I/O line.

If the selected port is part of simultaneous write ports (64-bit), this function will just modify the latched output value, but not update the output state of the I/O line.

There will be no effect if the I/O line is not configured as output port.

PARAMETERS

hdl

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

portNo

This value selects the port number that shall be used. Valid port numbers are 0 up to 15.

lineNo

This value selects the I/O line that shall be used. Valid line numbers are 0 up to 7.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/* reset I/O-line 5 of port 10 */
result = tpmc685ResetLine(hdl, 10, 5);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.3.7 tpmc685UpdateSimPorts

NAME

tpmc685UpdateSimPorts – update the selected 64-bit input and output ports in simultaneous mode

SYNOPSIS

```
TPMC685_STATUS tpmc685UpdateSimPorts
(
        TPMC685_HANDLE              hdl,
        unsigned int                simSelect
)
```

DESCRIPTION

Update selected 64-bit input and output ports. The function affects only ports configured for 64-bit (8-port) simultaneous input or output.

PARAMETERS

hdl

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

simSelect

This value specifies which of the 64-bit port groups and direction will be affected. The following definitions specify the 64-I/O lines (groups of 8 ports) and direction. The definitions can be ORed to select input and output of both groups within one call.

Definition	**Description**
TPMC685_SIMSEL_FRONTIN	Latch current input value of port 0 … port 7 (Front I/O ports)
TPMC685_SIMSEL_REARIN	Latch current input value of port 8 … port 15 (Rear I/O ports)
TPMC685_SIMSEL_FRONTOUT	Update output lines of port 0 … port 7 (Front I/O ports) with current output value
TPMC685_SIMSEL_REAROUT	Update output lines of port 8 … port 15 (Rear I/O ports) with current output value

EXAMPLE

```
#include "tpmc685api.h"

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/* Latch Front I/O input value and update Rear I/O output */
result = tpmc685UpdateSimPorts (hdl, TPMC685_SIMSEL_FRONTIN |
                                    TPMC685_SIMSEL_REAROUT);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.4 Watchdog Functions

3.4.1 tpmc685ConfigWatchdog

NAME

tpmc685ConfigWatchdog – configure output watchdog

SYNOPSIS

```
TPMC685_STATUS tpmc685ConfigWatchdog
(
        TPMC685_HANDLE          hdl,
        TPMC685_WDMODE          wdMode,
        TPMC685_TIMEBASE        wdBase,
        unsigned short          wdCount,
        unsigned char           wdSafePortStates[TPMC685_NUM_PORTS]
)
```

DESCRIPTION

This function configures the output watchdog function. The watchdog must be retriggered within the specified time. If the watchdog timer expires without a retrigger, the output lines will - depending on the configuration - freeze the current output state or set the outputs to the defined safe state.

PARAMETERS

hdl

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

wdMode

This value specifies the watchdog mode. The values are defined:

Mode	**Description**
TPMC685_WDMODE_FREEZE	When the watchdog expires, the state of the output lines will freeze. The watchdog is enabled afterwards.
TPMC685_WDMODE_SAFESTATE	When the watchdog expires the output lines will change to the defined safe state, which will be set in the parameter wd*SafePortStates*. The watchdog is enabled afterwards.
TPMC685_WDMODE_DISABLE	The watchdog will be disabled. All other parameters will be ignored.

wdBase

This value specifies the base clock frequency of the watchdog timer. Possible values are:

Watchdog Timer Base	**Description**
TPMC685_100NANOSEC	Timer base is 100ns
TPMC685_1MICROSEC	Timer base is 1µs
TPMC685_1MILLISEC	Timer base is 1ms

wdCount

This value specifies the count value of the timer. The watchdog preload value has a width of 16-bit.

wdSafePortStates

This parameter will be used if the watchdog mode is set to *TPMC685_WDMODE_SAFESTATE*. The parameter points to an array of *unsigned char* values with sixteen elements. The array defines the safe output state for all output ports, the value for input ports will be ignored. Element index 0 represents port 0, index 1 represents port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned char      safeStates[TPMC685_NUM_PORTS] = {0xFF, 0xFF, 0x00, 0x00,
                                 0x00, 0x00, 0x0F, 0x0F,
                                 0x11, 0x22, 0x44, 0x88,
                                 0x00, 0x00, 0x00, 0x00};

/*
** disable watchdog timer
*/
result = tpmc685ConfigWatchdog(  hdl,
                                 TPMC685_WDMODE_DISABLE,
                                 TPMC685_100NANOSEC,
                                 0,
                                 NULL);
if (result != TPMC685_OK)
{
     /* handle error */
}

…
```

```
…

/*
** enable watchdog timer (200µsec)
** – use safe output states
*/
result = tpmc685ConfigWatchdog(  hdl,
                                 TPMC685_WDMODE_SAFESTATE,
                                 TPMC685_1MICROSEC,
                                 200,
                                 safeStates);
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_BUSY	The Watchdog is already in expired state.

3.4.2 tpmc685TriggerWatchdog

NAME

tpmc685TriggerWatchdog – Trigger the output watchdog

SYNOPSIS

```
TPMC685_STATUS tpmc685TriggerWatchdog
(
    TPMC685_HANDLE          hdl
)
```

DESCRIPTION

Calling this function retriggers the watchdog timer.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/* Trigger (and reset) the watchdog timer */
result = tpmc685TriggerWatchdog(hdl);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_DISABLED	The watchdog timer is disabled.
TPMC685_ERR_WD_EXPIRED	The watchdog timer has expired.

3.5 Timer Functions

3.5.1 tpmc685ReadTimer

NAME

tpmc685ReadTimer – read the current value of a timer

SYNOPSIS

```
TPMC685_STATUS tpmc685ReadTimer
(
        TPMC685_HANDLE              hdl,
        unsigned int                timerNo,
        unsigned short              *timerValue
)
```

DESCRIPTION

Read the current value of a specified timer.

PARAMETERS

hdl

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

timerNo

This value selects the timer. The allowed values 0 and 1 correspond to the timer number.

timerValue

This parameter points to an *unsigned short* value where the timer’s current value will be stored.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned short     timerValue;

/* read the current value of timer 1 */
result = tpmc685ReadTimer(hdl, 1, &timerValue);
if (result == TPMC685_OK)
{
     printf(“Timer 1 - Count Value = 0x%04X\n”, timerValue);
}
else
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.5.2 tpmc685ConfigTimer

NAME

tpmc685ConfigTimer – configure timer

SYNOPSIS

```
TPMC685_STATUS tpmc685ConfigTimer
(
        TPMC685_HANDLE          hdl,
        unsigned int            timerNo,
        TPMC685_TIMER_MODE      timerMode,
        TPMC685_TIMEBASE        timerBase,
        unsigned short          timerCount
)
```

DESCRIPTION

This function configures the specified timer. The timer will not be started automatically after configuration. The timer must be started afterwards by calling the function *tpmc685StartTimer()*.

The timer must be disabled before calling this function. Use the function *tpmc685StopTimer()* to stop the timer.

PARAMETERS

hdl

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

timerNo

This value selects the timer. The allowed values 0 and 1 correspond to the timer number.

timerMode

This value specifies the timer mode. Possible values are:

Mode	Description
TPMC685_TIMER_MODSINGLE	Selects the Single Cycle Mode – the timer will stay in expired state when the specified time has expired. The timer must be stopped manually before starting the next cycle.
TPMC685_TIMER_MODCONT	Selects the Continuous Mode - the timer automatically reloads if the specified time has expired.

timerBase

This value specifies the base clock frequency for the timer. Possible values are:

Timer Base	**Description**
TPMC685_100NANOSEC	Timer base is 100ns
TPMC685_1MICROSEC	Timer base is 1µs
TPMC685_1MILLISEC	Timer base is 1ms

timerCount

This value specifies the count value of the timer. The preload value has a width of 16-bit.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/*
** setup timer 1 for continuous mode with a rate of 200ms (200*1ms)
*/
result = tpmc685ConfigTimer( hdl,
                             1,
                             TPMC685_TIMER_MODCONT,
                             TPMC685_1MILLISEC,
                             200);
if (result != TPMC685_OK)
{
    /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_BUSY	The timer is not in stopped state.

3.5.3 tpmc685StartTimer

NAME

tpmc685StartTimer – Start a timer

SYNOPSIS

```
TPMC685_STATUS tpmc685StartTimer
(
        TPMC685_HANDLE              hdl,
        unsigned int                timerNo,
        unsigned int                flags
)
```

DESCRIPTION

Start the specified timer. The specified timer must be configured using *tpmc685ConfigTimer()* before this function is called.

PARAMETERS

hdl

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

timerNo

This value selects the timer. The allowed values 0 and 1 corresponding to timer number.

flags

The flowing flags are defined and can be set ORed.

Flags	Description
TPMC685_ENABLE_EVENTS	Setting this flag enables event generation.

EXAMPLE

```
#include "tpmc685api.h"

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

…
```

…

```
/* Start Timer 1 */
result = tpmc685StartTimer(hdl, 1);
if (result != TPMC685_OK)
{
    /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_BUSY	The timer has already been started

3.5.4 tpmc685StopTimer

NAME

tpmc685StopTimer – Stop a timer

SYNOPSIS

```
TPMC685_STATUS tpmc685StopTimer
(
        TPMC685_HANDLE          hdl,
        unsigned int            timerNo
)
```

DESCRIPTION

Stop the specified timer

PARAMETERS

hdl

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

timerNo

This value selects the timer. The allowed values 0 and 1 correspond to the timer number.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/* Stop Timer 1 */
result = tpmc685StopTimer(hdl, 1);
if (result != TPMC685_OK)
{
     /* Handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.6 Event Functions

3.6.1 tpmc685EnableDigitalEvents

NAME

tpmc685EnableDigitalEvents – enable event generation for specified I/O-lines

SYNOPSIS

```
TPMC685_STATUS tpmc685EnableDigitalEvents
(
      TPMC685_HANDLE          hdl,
      unsigned char           risingEdge[TPMC685_NUM_PORTS],
      unsigned char           fallingEdge[TPMC685_NUM_PORTS]
)
```

DESCRIPTION

This function enables the event generation on the specified I/O lines. After calling this function the specified events will be detected and their occurrence will be stored.

If this function enables the event detection for a specified I/O line and the waiting function is called later, the function will return if the specified event has been detected before the waiting function has been called.

If this function enabled the event detection after the waiting function has been called, the first occurrence of the event after calling this function will unblock the waiting function.

PARAMETERS

hdl

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

risingEdge

This parameter contains an array of *unsigned char* values with sixteen elements. Each array element represents one I/O port. A set bit in the array element specifies that events shall be generated if rising edges (low-to-high transition) are detected on the corresponding I/O-line. Element index 0 represents the mask for port 0, index 1 represents the mask for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

fallingEdge

This parameter contains an array of *unsigned char* values with sixteen elements. Each array element represents one I/O port. A set bit in the array element specifies that events shall be generated if falling edges (high-to-low transition) are detected on the corresponding I/O-line. Element index 0 represents the mask for port 0, index 1 represents the mask for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned char      risingEdge[TPMC685_NUM_PORTS] =
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned char      fallingEdge[TPMC685_NUM_PORTS] =
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/*
** enable edge detection on
**    - falling edges on port 0 – I/O-line 3
**    - any edge on port 0 – I/O-line 5
**    - rising edges on port 5 – I/O-line 0
*/
fallingEdge[0] |= (1 << 3);
fallingEdge[0] |= (1 << 5);
risingEdge[0]  |= (1 << 5);
risingEdge[5]  |= (1 << 0);
result = tpmc685EnableDigitalEvents ( hdl,
                                      risingEdge,
                                      fallingEdge);
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.6.2 tpmc685DisableDigitalEvents

NAME

tpmc685DisableDigitalEvents – disable event generation for specified I/O-lines

SYNOPSIS

```
TPMC685_STATUS tpmc685DisableDigitalEvents
(
      TPMC685_HANDLE           hdl,
      unsigned char            risingEdge[TPMC685_NUM_PORTS],
      unsigned char            fallingEdge[TPMC685_NUM_PORTS]
)
```

DESCRIPTION

This function disables the event generation on the specified I/O lines

PARAMETERS

hdl

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

risingEdge

This parameter contains an array of *unsigned char* values with sixteen elements. Each array element represents one I/O port. A set bit in the array element specifies that event generation shall be stopped if rising edges (low-to-high transition) are detected on the corresponding I/O-line. Element index 0 represents the mask for port 0, index 1 represents the mask for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

fallingEdge

This parameter contains an array of *unsigned char* values with sixteen elements. Each array element represents one I/O port. A set bit in the array element specifies that event generation shall be stopped if falling edges (high-to-low transition) are detected on the corresponding I/O-line. Element index 0 represents the mask for port 0, index 1 represents the mask for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned char      risingEdge[TPMC685_NUM_PORTS] =
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned char      fallingEdge[TPMC685_NUM_PORTS] =
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/*
** disable edge detection on
**     - falling edges on port 0 – I/O-line 3
**     - any edge on port 0 – I/O-line 5
**     - rising edges on port 5 – I/O-line 0
*/
fallingEdge[0] |= (1 << 3);
fallingEdge[0] |= (1 << 5);
risingEdge[0]  |= (1 << 5);
risingEdge[5]  |= (1 << 0);
result = tpmc685DisableDigitalEvents( hdl,
                                      risingEdge,
                                      fallingEdge);
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.

3.6.3 tpmc685WaitIoLineEvent

NAME

tpmc685WaitIoLineEvent – wait for a specified event on a specified digital I/O line

SYNOPSIS

```
TPMC685_STATUS tpmc685WaitIoLineEvent
(
        TPMC685_HANDLE              hdl,
        unsigned int                portNo,
        unsigned int                lineNo,
        TPMC685_TRANSITION          transition,
        int                         timeout
)
```

DESCRIPTION

Wait for a specified transition on the selected digital input line. If the specified event occurs the function returns, if the event does not occur the function returns after a specified timeout time returning an error code.

PARAMETERS

hdl

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

portNo

This value selects the port number that shall be used. Valid port numbers are 0 up to 15.

lineNo

This value selects the I/O line that shall be used. Valid line numbers are 0 up to 7.

transition

This value specifies the transition on which the specified event should occur.

Value	**Description**
TPMC685_RISINGEDGE	Wait for a rising edge, low-to-high transition
TPMC685_FALLINGEDGE	Wait for a falling edge, high-to-low transition
TPMC685_ANYEDGE	Wait for any edge, any transition

timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TPMC685_WAIT_FOREVER must be specified.

EXAMPLE

```
#include "tpmc685api.h"

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/*
** wait for a rising edge on digital input of port 1, I/O-line 4
** timeout after 5 seconds
*/
result = tpmc685WaitIoLineEvent( hdl,
                                 1,
                                 4,
                                 TPMC685_RISINGEDGE,
                                 5000);
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_TIMEOUT	The event has not occurred within the specified time.

3.6.4 tpmc685WaitDigitalEvents

NAME

tpmc685WaitDigitalEvents – wait for an event on specified I/O-lines

SYNOPSIS

```
TPMC685_STATUS tpmc685WaitDigitalEvents
(
        TPMC685_HANDLE          hdl,
        unsigned char           risingEdge[TPMC685_NUM_PORTS],
        unsigned char           fallingEdge[TPMC685_NUM_PORTS],
        int                     timeout,
        unsigned char           risingEvents[TPMC685_NUM_PORTS],
        unsigned char           fallingEvents[TPMC685_NUM_PORTS]
)
```

DESCRIPTION

Wait for specified transitions on the specified digital input lines. If at least one of the specified events occurs the function returns. If the event does not occur, the function returns after the specified timeout time returning an error code.

PARAMETERS

hdl

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

risingEdge

This parameter contains an array of *unsigned char* values with sixteen elements. Each array element represents one I/O port. A set bit in the array element specifies that an event shall be generated if rising edge (low-to-high transition) is detected on the corresponding I/O-line. Element index 0 represents the mask for port 0, index 1 represents the mask for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

fallingEdge

This parameter contains an array of *unsigned char* values with sixteen elements. Each array element represents one I/O port. A set bit in the array element specifies that an event shall be generated if falling edge (high-to-low transition) is detected on the corresponding I/O-line. Element index 0 represents the mask for port 0, index 1 represents the mask for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.

timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TPMC685_WAIT_FOREVER must be specified.

risingEvents

This buffer indicates the occurred rising edge events. Element index 0 represents the occurred events for port 0, index 1 represents the occurred events for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.
If a NULL pointer is specified, the occurred events will not be returned.

fallingEvents

This buffer indicates the occurred falling edge events. Element index 0 represents the occurred events for port 0, index 1 represents the occurred events for port 1, and so on. Bit-0 of each element represents I/O Line 0 of the port, bit-1 represents I/O Line 1 of the port, and so on.
If a NULL pointer is specified, the occurred events will not be returned.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;
unsigned char      risingEdge[TPMC685_NUM_PORTS] =
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned char      fallingEdge[TPMC685_NUM_PORTS] =
                        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned char      risingEvents[TPMC685_NUM_PORTS];
unsigned char      fallingEvents[TPMC685_NUM_PORTS];

…
```

```
…

/*
** wait for one of the following events:
**    - falling edge on port 0 – I/O-line 3
**    - any edge on port 0 – I/O-line 5
**    - rising edge on port 5 – I/O-line 0
** timeout after 5 seconds
*/
fallingEdge[0] |= (1 << 3);
fallingEdge[0] |= (1 << 5);
risingEdge[0]  |= (1 << 5);
risingEdge[5]  |= (1 << 0);
result = tpmc685WaitDigitalEvents(    hdl,
                                      risingEdge,
                                      fallingEdge,
                                      5000,
                                      risingEvents,
                                      fallingEvents);
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_TIMEOUT	The event has not occurred within the specified time.

3.6.5 tpmc685WaitTimerEvent

NAME

tpmc685WaitTimerEvent – wait for a timer event on a specified timer

SYNOPSIS

```
TPMC685_STATUS tpmc685WaitTimerEvent
(
        TPMC685_HANDLE              hdl,
        unsigned int                timerNo,
        int                         timeout
)
```

DESCRIPTION

Wait for a timer event on the specified timer. The function returns to the caller if a timer event occurs. If the event does not occur, the function returns after the specified timeout time returning an error code.

PARAMETERS

hdl

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

timerNo

This value selects the timer. The allowed values 0 and 1 correspond to the timer number.

timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TPMC685_WAIT_FOREVER must be specified.

EXAMPLE

```
#include “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

/*
** wait for an event on timer 0
*/
result = tpmc685WaitTimerEvent(  hdl,
                                 0,
                                 TPMC685_WAIT_FOREVER );
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC685_ERR_INVAL	An invalid parameter value has been specified.
TPMC685_ERR_TIMEOUT	The event has not occurred within the specified time.

3.6.6 tpmc685WaitWatchdogEvent

NAME

tpmc685WaitWatchdogEvent – wait for a watchdog expired event

SYNOPSIS

```
TPMC685_STATUS tpmc685WaitWatchdogEvent
(
    TPMC685_HANDLE          hdl,
    int                     timeout
)
```

DESCRIPTION

The function waits until the watchdog timer expires. if the event does not occur, the function returns after a specified timeout time returning an error code.

PARAMETERS

hdl

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

timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TPMC685_WAIT_FOREVER must be specified.

EXAMPLE

```
#include  “tpmc685api.h”

TPMC685_HANDLE     hdl;
TPMC685_STATUS     result;

…
```

…

```
/*
** wait for a watchdog interrupt
** - timeout after 5 seconds
*/
result = tpmc685WaitWatchdogEvent(    hdl,
                                      5000 );
if (result != TPMC685_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC685_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC685_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC685_ERR_INVAL	An invalid parameter value has been specified
TPMC685_ERR_TIMEOUT	The event has not occurred within the specified time

4 Diagnostic

If the TPMC685 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 display information of a correct running TPMC685 driver (see also the proc man pages).

```
# lspci -v
  …
04:01.0 Non-VGA unclassified device: TEWS Technologies GmbH Device 02ad
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: bus master, medium devsel, latency 64, IRQ 16
        Memory at feb9fc00 (32-bit, non-prefetchable) [size=256]
        Kernel driver in use: TEWS TECHNOLOGIES TPMC685 16x8 Digital IO
        Kernel modules: tpmc685drv
  …
```

```
# cat /proc/devices
Character devices:
  1 mem
  …
248 tpmc685drv
  …
```

```
# cat /proc/iomem
00000000-0000ffff : reserved
00010000-0009fbff : System RAM
  …
  feb00000-febfffff : PCI Bus 0000:04
    feb9fc00-feb9fcff : 0000:04:01.0
      feb9fc00-feb9fcff : TPMC685
  …
```