

*The Embedded I/O Company*



---

# TPMC680-SW-25

## Integrity Device Driver

8 x 8 Bit Digital I/O

Version 1.0.x

## User Manual

Issue 1.0.0

September 2012



Ehlbeek 15a  
30938 Burgwedel  
fon 05139-9980-0  
fax 05139-9980-49

[www.powerbridge.de](http://www.powerbridge.de)  
[info@powerbridge.de](mailto:info@powerbridge.de)

---

### EWS TECHNOLOGIES GmbH

.m Bahnhof 7 25469 Halstenbek, Germany  
4101 4058 0 Fax: +49 (0) 4101 4058 19  
o@tews.com [www.tews.com](http://www.tews.com)

## TPMC680-SW-25

Integrity Device Driver

8 x 8 Bit Digital I/O

Supported Modules:  
TPMC680-10

This document contains information, which is proprietary to TEWS TECHNOLOGIES GmbH. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES GmbH reserves the right to change the product described in this document at any time without notice.

TEWS TECHNOLOGIES GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2012 by TEWS TECHNOLOGIES GmbH

| Issue | Description | Date               |
|-------|-------------|--------------------|
| 1.0.0 | First Issue | September 10, 2012 |

# Table of Contents

|          |                               |           |
|----------|-------------------------------|-----------|
| <b>1</b> | <b>INTRODUCTION.....</b>      | <b>4</b>  |
| <b>2</b> | <b>INSTALLATION.....</b>      | <b>5</b>  |
|          | 2.1 Driver Installation.....  | 5         |
|          | 2.2 TPMC680 Applications..... | 5         |
| <b>3</b> | <b>API DOCUMENTATION.....</b> | <b>6</b>  |
|          | 3.1 tpmc680Open.....          | 6         |
|          | 3.2 tpmc680Close.....         | 8         |
|          | 3.3 tpmc680SetPortMode.....   | 10        |
|          | 3.4 tpmc680ReadPort.....      | 14        |
|          | 3.5 tpmc680WritePort.....     | 16        |
|          | 3.6 tpmc680ReadPort64.....    | 18        |
|          | 3.7 tpmc680WritePort64.....   | 20        |
|          | 3.8 tpmc680Receive16.....     | 22        |
|          | 3.9 tpmc680Send16.....        | 24        |
|          | 3.10 tpmc680Receive32.....    | 26        |
|          | 3.11 tpmc680Send32.....       | 28        |
|          | 3.12 tpmc680WaitForEvent..... | 30        |
| <b>4</b> | <b>APPENDIX.....</b>          | <b>32</b> |
|          | 4.1 Example Applications..... | 32        |
|          | 4.1.1 tpmc680_in_8.c.....     | 32        |
|          | 4.1.2 tpmc680_out_8.c.....    | 32        |
|          | 4.1.3 tpmc680_in_16.c.....    | 32        |
|          | 4.1.4 tpmc680_out_16.c.....   | 32        |
|          | 4.1.5 tpmc680_in_32.c.....    | 33        |
|          | 4.1.6 tpmc680_out_32.c.....   | 33        |
|          | 4.1.7 tpmc680_in_64.c.....    | 33        |
|          | 4.1.8 tpmc680_out_64.c.....   | 33        |
|          | 4.1.9 tpmc680_event.c.....    | 34        |

# 1 Introduction

The TPMC680-SW-25 Integrity device driver software allows the operation of the supported PMC conforming to the Integrity I/O system specification.

The driver software uses mutual exclusion to prevent simultaneous requests by multiple tasks from interfering with each other.

The TPMC680-SW-25 device driver supports the following features:

- direct reading for input ports (8 bit / 64 bit)
- direct writing for output ports (8 bit / 64 bit)
- buffered read for input ports (16/32 bit handshake mode)
- buffered write for output ports (16/32 bit handshake mode)
- configuring ports
- wait for a specified input event (8 bit / 64 bit ports)

The TPMC680-SW-25 supports the modules listed below:

|            |   |       |
|------------|---|-------|
| TPMC680-10 | 8 x 8 Bit Digital Inputs/Outputs (5V TTL) | (PMC) |
|------------|---|-------|

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

|                            |
|----------------------------|
| TPMC680 User Manual        |
| TPMC680 Engineering Manual |

---

## 2 Installation

The following files are located on the distribution media:

Directory path TPMC680-SW-25:

|                         |   |
|-------------------------|---|
| tpmc680.c               | TPMC680 device driver source                    |
| tpmc680def.h            | TPMC680 driver include file                     |
| tpmc680.h               | TPMC680 include file for driver and application |
| tpmc680api.c            | Application interface, simplifies device access |
| tpmc680api.h            | Include file for API and applications           |
| example/*.c             | Path with example applications                  |
| TPMC680-SW-25-1.0.0.pdf | PDF copy of this manual                         |
| ChangeLog.txt           | Release history                                 |
| Release.txt             | Release information                             |

### 2.1 Driver Installation

Copy the TPMC680 driver files (tpmc680.c, tpmc680.h, tpmc680api.h, and tpmc680.h) into a desired driver or project path. The driver source file tpmc680.c must be included into the kernel project and the BSP path must be added to the include search paths of the file. (Set Options... → Project → Include Directories, than double click and add a new path and select ../int680/bsp)

Afterwards the project must be rebuilt. The driver will be started automatically after booting the image and the driver will be requested if a matching device is detected in the system.

For further information about building a kernel, please refer to MULTI and INTEGRITY Documentation.

### 2.2 TPMC680 Applications

Copy the TPMC680 API files (tpmc680api.c, tpmc680api.h, and tpmc680.h) into a desired application path, and include tpmc680api.c into the application project.

The application source file must include tpmc680api.h. If these steps are done, the TPMC680 API can be used and the devices will be accessible.

## **3 API Documentation**

### **3.1 tpmc680Open**

#### **NAME**

tpmc680Open() – open a device.

#### **SYNOPSIS**

```
TPMC680_HANDLE tpmc680Open  
(  
    char                *DeviceName  
)
```

#### **DESCRIPTION**

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

#### **PARAMETERS**

##### *DeviceName*

This parameter points to a null-terminated string that specifies the name of the device. The first TPMC680 device is named “tpmc680\_0”, the second device is named “tpmc680\_1” and so on.

#### **EXAMPLE**

```
#include "tpmc680api.h"  
  
TPMC680_HANDLE    hdl;  
  
/*  
** open file descriptor to device  
*/  
hdl = tpmc680Open("tpmc680_0");  
if (hdl == NULL)  
{  
    /* handle open error */  
}
```

## **RETURNS**

A device descriptor pointer or NULL if the function fails.

## 3.2 tpmc680Close

### NAME

tpmc680Close() – close a device.

### SYNOPSIS

```
TPMC680_STATUS tpmc680Close  
(  
    TPMC680_HANDLE      hdl  
)
```

### DESCRIPTION

This function closes previously opened devices.

### PARAMETERS

*hdl*

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

### EXAMPLE

```
#include "tpmc680api.h"  
  
TPMC680_HANDLE      hdl;  
TPMC680_STATUS      result;  
  
/*  
** close the device  
*/  
result = tpmc680Close(hdl);  
if (result != TPMC680_OK)  
{  
    /* handle close error */  
}
```



## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description                            |
|----------------------------|--|
| TPMC680_ERR_INVALID_HANDLE | The specified device handle is invalid |

### 3.3 tpmc680SetPortMode

#### NAME

tpmc680SetPortMode – Configure port

#### SYNOPSIS

```
TPMC680_STATUS tpmc680SetPortMode
(
    TPMC680_HANDLE          hdl,
    unsigned int             portNo,
    unsigned int             portSize,
    unsigned int             portDirection,
    unsigned int             handshakeMode,
    unsigned int             handshakeFifoLevelMode
)
```

#### DESCRIPTION

This function configures the specified port of the TPMC680. The function sets size, direction and handshake modes. If port sizes greater 8 bit are used some (hardware) ports will be concatenated to a (software) port which is responsible to control the I/O function. Mainly responsible for port concatenations are port 0 and 2. Port 0 can be used for 16 and 32 bit handshake and 64 bit synchronous I/O. Port 2 can be used for 16 bit handshake I/O.

The table below shows to which port number the (hardware) ports will be assigned at the possible configurations of ports 0 and 2.

| (Hardware) Port        | 7                           | 6            | 5            | 4            | 3                  | 2            | 1                  | 0            |
|------------------------|-----------------------------|--------------|--------------|--------------|--------------------|--------------|--------------------|--------------|
| (Software) Port number | 7<br>(8 bit)                | 6<br>(8 bit) | 5<br>(8 bit) | 4<br>(8 bit) | 3<br>(8 bit)       | 2<br>(8 bit) | 1<br>(8 bit)       | 0<br>(8 bit) |
|                        | 7<br>(8 bit)                | 6<br>(8 bit) | 5<br>(8 bit) | 4<br>(8 bit) | 3<br>(8 bit)       | 2<br>(8 bit) | 0<br>(16 bit / HS) |              |
|                        | 7<br>(8 bit)                | 6<br>(8 bit) | 5<br>(8 bit) | 4<br>(8 bit) | 2<br>(16 bit / HS) |              | 1<br>(8 bit)       | 0<br>(8 bit) |
|                        | 7<br>(8 bit)                | 6<br>(8 bit) | 5<br>(8 bit) | 4<br>(8 bit) | 2<br>(16 bit / HS) |              | 0<br>(16 bit / HS) |              |
|                        | 7<br>(8 bit)                | 6<br>(8 bit) | 5<br>(8 bit) | 4<br>(8 bit) | 0<br>(32 bit / HS) |              |                    |              |
|                        | 0<br>(64 bit / synchronous) |              |              |              |                    |              |                    |              |

Additionally to the port concatenations the direction of port 4 and port 5 may be changed if port 0 or port 2 is used in handshake mode. Port 4 will be configured as input port and port 5 may be configured for output. Bit 0 and 1 will be reserved for the handshake signals and are not anymore controlled by the ports.

**Please also refer to the TPMC680 User Manual to get more information about the port configuration and use signals.**

**Changing a port size from a bigger to a smaller size will also change the mode of the connected ports. The ports will be set to 8 bit mode and they will keep the configured direction.**

## PARAMETERS

### *hdl*

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

### *portNo*

This argument specifies the port that shall be configured. Valid values are between 0 and 7.

### *portSize*

This argument specifies the size of the port. The following table describes the allowed port sizes and for which ports they are allowed.

| Value                   | Ports                  | Description   |
|-------------------------|------------------------|---|
| TPMC680_MODE_SIZE_8BIT  | 0, 1, 2, 3, 4, 5, 6, 7 | The port has a width of 8 bit. Each port can be accessed separately.  |
| TPMC680_MODE_SIZE_16BIT | 0,2                    | The port has a width of 16 bit and the output is controlled by the handshake signals. Two ports are used together. If port 0 is selected port 1 is used also. If port 2 is selected also port 3 will be used. The configuration of the connected ports is always adapted.<br><b>If this mode is selected for any port the handshake port 4 will be configured as an 8-bit input port.</b> |
| TPMC680_MODE_SIZE_32BIT | 0                      | The port has a width of 32 bit and the output is controlled by the handshake signals. The ports 0, 1, 2 and 3 will be used together. The configuration of the connected ports is always set together.<br><b>If this mode is selected the handshake port 4 will be configured as an 8-bit input port.</b>  |
| TPMC680_MODE_SIZE_64BIT | 0                      | All ports are connected and can be used as simple 64 bit input or output port. All ports get the same configuration.  |

*portDirection*

This argument specifies the direction of the port. All connected ports will be set to the same direction. Allowed values are:

| Value                   | Description                              |
|-------------------------|--|
| TPMC680_MODE_DIR_INPUT  | The port will be used as an input port.  |
| TPMC680_MODE_DIR_OUTPUT | The port will be used as an output port. |

*handshakeMode*

This argument specifies the handshake mode and is only valid if the port is configured for 16 or 32 bit mode (*TPMC680\_MODE\_SIZE\_16BIT*, *TPMC680\_MODE\_SIZE\_32BIT*). Using an output handshake, will change the direction of port 5 to output. The allowed values are:

| Value                           | Description   |
|---------------------------------|---|
| TPMC680_MODE_HSFLAG_NO          | No output handshake will be used.                   |
| TPMC680_MODE_HSFLAG_INTERLOCKED | The interlocked output handshake mode will be used. |
| TPMC680_MODE_HSFLAG_PULSED      | The pulsed output handshake mode will be used.      |

*handshakeFifoLevelMode*

This argument specifies the handshake event depending on the handshake FIFO fill level. This value is only used if an output handshake is configured. Allowed values are:

| Value                         | Description                           |
|-------------------------------|---------------------------------------|
| TPMC680_MODE_HSFIFOEV_NOTFULL | The event announces FIFO is not full. |
| TPMC680_MODE_HSFIFOEV_EMPTY   | The event announces FIFO is empty.    |

## EXAMPLE

```
#include "tpmc680api.h"

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;

/*
** Configure port (2)
**     Size: 16-bit
**     Direction: output
**     handshake: interlocked / output event on empty FIFO
*/
result = tpmc680SetPortMode (    hdl,
                                2,
                                TPMC680_MODE_SIZE_16BIT,
                                TPMC680_MODE_DIR_OUTPUT,
                                TPMC680_MODE_HSFLAG_INTERLOCKED,
                                TPMC680_MODE_HSFIFOEV_EMPTY);

if (result != TPMC680_OK)
{
    /* handle error */
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description                                      |
|----------------------------|--|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.         |
| TPMC680_ERR_INVALID        | An argument contains an invalid value.           |
| TPMC680_ERR_CHRNG          | An invalid port number has been specified.       |
| TPMC680_ERR_ACCESS         | The specified port configuration is not allowed. |

## 3.4 tpmc680ReadPort

### NAME

tpmc680ReadPort – Read state of 8-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680ReadPort  
(  
    TPMC680_HANDLE          hdl,  
    unsigned int            portNo,  
    unsigned char           *pPortVal  
)
```

### DESCRIPTION

This function reads the current state of the input lines of an 8 bit port on the TPMC680.

**The port must be configured in 8 bit mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*portNo*

This argument specifies the port that shall be read. Valid values are between 0 and 7.

*pPortVal*

This pointer points to an unsigned char where the current state of the port will be stored to.

## EXAMPLE

```
#include "tpmc680api.h"

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;
unsigned char      portState;

/*
** Read from 8-bit port (2)
*/
result = tpmc680ReadPort (  hdl,
                           2,
                           &portState);

if (result == TPMC680_OK)
{
    printf("Port2: 0x%02X\n", portState);
}
else
{
    /* handle error */
}

```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_INVAL          | A specified pointer is NULL.                        |
| TPMC680_ERR_CHRNG          | An invalid port number has been specified.          |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.5 tpmc680WritePort

### NAME

tpmc680WritePort – Write new output value to 8-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680WritePort
(
    TPMC680_HANDLE          hdl,
    unsigned int             portNo,
    unsigned char            portVal
)
```

### DESCRIPTION

This function writes a new output value to an 8 bit port of the TPMC680.

**The port must be configured in 8 bit output mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*portNo*

This argument specifies the port that shall be read. Valid values are between 0 and 7.

*portVal*

This argument specifies the new output value.



## EXAMPLE

```
#include "tpmc680api.h"

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;

/*
** Set 8-bit port (2) (new value 12(hex))
*/
result = tpmc680WritePort ( hdl,
                            2,
                            0x12);

if (result != TPMC680_OK)
{
    /* handle error */
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_CHRNG          | An invalid port number has been specified.          |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.6 tpmc680ReadPort64

### NAME

tpmc680ReadPort64 – Read state of 64-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680ReadPort64
(
    TPMC680_HANDLE          hdl,
    unsigned int             *pPortVal0_31,
    unsigned int             *pPortVal32_63
)
```

### DESCRIPTION

This function reads the current state of the input lines of the 64 bit port on the TPMC680.

**The port must be configured in 64 bit mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*pPortVal0\_31*

This pointer points to an unsigned int (32-bit) where the current state of the ports 0...3 will be stored. Port 0 will be stored to bits 0...7, Port 1 to bits 8...15, and so on.

*pPortVal32\_63*

This pointer points to an unsigned int (32-bit) where the current state of the ports 4...7 will be stored. Port 4 will be stored to bits 0...7, Port 5 to bits 8...15, and so on.

## EXAMPLE

```
#include "tpmc680api.h"

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;
unsigned int       portStateLow;
unsigned int       portStateHigh;

/*
** Read from 64-bit port
*/
result = tpmc680ReadPort64( hdl,
                            &portStateLow,
                            &portStateHigh);

if (result == TPMC680_OK)
{
    printf("Port7..0: 0x%08X%08X\n", portStateHigh, portStateLow);
}
else
{
    /* handle error */
}

```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_INVALID        | A specified pointer is NULL.                        |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.7 tpmc680WritePort64

### NAME

tpmc680WritePort64 – Write new output value to 64-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680WritePort64
(
    TPMC680_HANDLE          hdl,
    unsigned int             portVal0_31,
    unsigned int             portVal32_63
)
```

### DESCRIPTION

This function writes a new output value to the 64 bit port of the TPMC680.

**The port must be configured in 64 bit output mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*portVal0\_31*

This argument specifies the new output value of the ports 0...3. Port 0 corresponds to bits 0...7, Port 1 to bits 8...15, and so on.

*portVal32\_63*

This argument specifies the new output value of the ports 4...7. Port 4 corresponds to bits 0...7, Port 5 to bits 8...15, and so on.

## EXAMPLE

```
#include "tpmc680api.h"

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;

/*
** Set 64-bit port (new value 7766554433221100 (hex))
*/
result = tpmc680WritePort64 (    hdl,
                                0x33221100,
                                0x77665544);

if (result != TPMC680_OK)
{
    /* handle error */
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.8 tpmc680Receive16

### NAME

tpmc680Receive16 – Read data received on 16-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680Receive16
(
    TPMC680_HANDLE          hdl,
    unsigned int            portNo,
    unsigned int            bufSize,
    unsigned short          *pBuf,
    unsigned int            *pValidData
)
```

### DESCRIPTION

This function reads data that has been received on a 16 bit input port of the TPMC680.

**The port must be configured in 16 bit input mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*portNo*

This argument specifies the port that shall be read. Valid values are 0 and 2.

*bufSize*

This argument specifies the number data words (16 bit) which can be copied into the input buffer.

*pBuf*

This pointer points to the input buffer where the received data will be stored.

*pValidData*

This pointer points to an unsigned int value where the number of received (valid) data values will be stored.

## EXAMPLE

```
#include "tpmc680api.h"

#define BUFSIZE      5

TPMC680_HANDLE      hdl;
TPMC680_STATUS      result;
unsigned short       inBuf[BUFSIZE];
unsigned int         numData;

/*
** Read received data from 16-bit port (2)
*/
result = tpmc680Receive16 ( hdl,
                           2,
                           BUFSIZE,
                           inBuf,
                           &numData);

if (result == TPMC680_OK)
{
    for (i = 0; i < numData; i++)
        printf("[%d] 0x%04X\n", i, inBuf[i]);
}
else
{
    /* handle error */
}
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_INVAL          | A specified pointer is NULL.                        |
| TPMC680_ERR_CHRNG          | An invalid port number has been specified.          |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.9 tpmc680Send16

### NAME

tpmc680Send16 – Send data on 16-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680Send16
(
    TPMC680_HANDLE          hdl,
    unsigned int            portNo,
    unsigned int            bufSize,
    unsigned short          *pBuf,
    unsigned int            *pSentData
)
```

### DESCRIPTION

This function sends data on a 16 bit port of the TPMC680. The function transfers the data into a FIFO and starts transmission. It will not wait until data is sent.

**The port must be configured in 16 bit output mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*portNo*

This argument specifies the port that shall be used. Valid values are 0 and 2.

*bufSize*

This argument specifies the number data words (16 bit) in the output buffer.

*pBuf*

This pointer points to the output buffer containing the data ready to send.

*pSentData*

This pointer points to an unsigned int value where the number of successfully transferred data values will be stored.



## EXAMPLE

```
#include "tpmc680api.h"

#define BUFSIZE      5

TPMC680_HANDLE      hdl;
TPMC680_STATUS      result;
unsigned short       outBuf[BUFSIZE] = {0x1111,0x2222,0x3333,0x4444,0x5555};
unsigned int         numData;

/*
** Read received data from 16-bit port (2)
*/
result = tpmc680Send16 (    hdl,
                          2,
                          BUFSIZE,
                          outBuf,
                          &numData);

if (result != TPMC680_OK)
{
    /* handle error */
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_INVAL          | A specified pointer is NULL.                        |
| TPMC680_ERR_CHRNG          | An invalid port number has been specified.          |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.10tpmc680Receive32

### NAME

tpmc680Receive32 – Read data received on 32-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680Receive32
(
    TPMC680_HANDLE          hdl,
    unsigned int             bufSize,
    unsigned int             *pBuf,
    unsigned int             *pValidData
)
```

### DESCRIPTION

This function reads data that has been received on the 32 bit input port of the TPMC680.

**The port must be configured in 32 bit input mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*bufSize*

This argument specifies the number data words (32 bit) which can be copied into the input buffer.

*pBuf*

This pointer points to the input buffer where the received data will be stored.

*pValidData*

This pointer points to an unsigned int value where the number of received (valid) data values will be stored.

## EXAMPLE

```
#include "tpmc680api.h"

#define BUFSIZE    5

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;
unsigned int       inBuf[BUFSIZE];
unsigned int       numData;

/*
** Read received data from 32-bit port (2)
*/
result = tpmc680Receive32 ( hdl,
                           BUFSIZE,
                           inBuf,
                           &numData);

if (result == TPMC680_OK)
{
    for (i = 0; i < numData; i++)
        printf("[%d] 0x%08X\n", i, inBuf[i]);
}
else
{
    /* handle error */
}

```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_INVAL          | A specified pointer is NULL.                        |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.11 tpmc680Send32

### NAME

tpmc680Send32 – Send data on 32-bit port

### SYNOPSIS

```
TPMC680_STATUS tpmc680Send32
(
    TPMC680_HANDLE          hdl,
    unsigned int             bufSize,
    unsigned int             *pBuf,
    unsigned int             *pSentData
)
```

### DESCRIPTION

This function sends data on the 32 bit port of the TPMC680. The function transfers the data into a FIFO and starts transmission. It will not wait until data is sent.

**The port must be configured in 32 bit output mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*bufSize*

This argument specifies the number data words (32 bit) in the output buffer.

*pBuf*

This pointer points to the output buffer containing the data ready to send.

*pSentData*

This pointer points to an unsigned int value where the number of successfully transferred data values will be stored.

## EXAMPLE

```
#include "tpmc680api.h"

#define BUFSIZE      3

TPMC680_HANDLE      hdl;
TPMC680_STATUS      result;
unsigned int         outBuf[BUFSIZE] = {0x11112222,0x33334444,0x55556666};
unsigned int         numData;

/*
** Send data on 32-bit port
*/
result = tpmc680Send32 (    hdl,
                          BUFSIZE,
                          outBuf,
                          &numData);

if (result != TPMC680_OK)
{
    /* handle error */
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description   |
|----------------------------|---|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.            |
| TPMC680_ERR_INVAL          | A specified pointer is NULL.                        |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration. |

## 3.12tpmc680WaitForEvent

### NAME

tpmc680WaitForEvent – Wait for a specified input event

### SYNOPSIS

```
TPMC680_STATUS tpmc680WaitForEvent
(
    TPMC680_HANDLE          hdl,
    unsigned int            portNo,
    unsigned int            lineNo,
    unsigned int            transition,
    unsigned int            timeout
)
```

### DESCRIPTION

This function waits for a specified event on a specified input line of the TPMC680.

**The port must be configured in 8 bit or 64 bit input mode, otherwise the function will fail.**

### PARAMETERS

*hdl*

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

*portNo*

This argument specifies the port. Valid values are between 0 and 7.

*lineNo*

This argument specifies the port's line number. Valid values are between 0 and 7.

*transition*

This argument specifies the transition event to wait for. The following events are supported:

| Value               | Description  |
|---------------------|--|
| TPMC680_IO_EDGE_HI  | The event will occur if the specified input line changes from Low to High. |
| TPMC680_IO_EDGE_LO  | The event will occur if the specified input line changes from High to Low. |
| TPMC680_IO_EDGE_ANY | The event will occur if the specified input line changes its value.        |

### *timeout*

This argument specifies the timeout in milliseconds. If the specified event does not occur within the specified time, the function will return with an error code. The timeout granularity is in seconds.

## EXAMPLE

```
#include "tpmc680api.h"

TPMC680_HANDLE    hdl;
TPMC680_STATUS    result;

/*
** Wait for a high to low transition on line 5 of port 6
**     Timeout after 10000 milliseconds
*/
result = tpmc680WaitForEvent (    hdl,
                                6,
                                5,
                                TPMC680_IO_EDGE_LO,
                                10000);

if (result != TPMC680_OK)
{
    /* handle error */
}
```

## RETURNS

On success, TPMC680\_OK is returned. In the case of an error, the appropriate error code is returned by the function.

## ERROR CODES

| Error Code                 | Description  |
|----------------------------|--|
| TPMC680_ERR_INVALID_HANDLE | The specified TPMC680_HANDLE is invalid.   |
| TPMC680_ERR_INVAL          | A specified argument contains an invalid value.                                  |
| TPMC680_ERR_CHRNG          | An invalid port number has been specified.                                       |
| TPMC680_ERR_ACCESS         | Access not allowed with current port configuration.                              |
| TPMC680_ERR_BUSY           | There is already an active job waiting for an event on the specified input line. |
| TPMC680_ERR_TIMEOUT        | The function timed out   |

---

# **4 Appendix**

## **4.1 Example Applications**

The example application shall give an overview about the use of the TPMC680 devices and how to use the TPMC680 API.

### **4.1.1 tpmc680\_in\_8.c**

This simple example configures the ports for 8-bit input and executes reads for all input ports on all installed TPMC680 boards.

Program flow:

- open devices
- configure all ports for 8-bit input
- periodically execute read on all ports
- close devices

### **4.1.2 tpmc680\_out\_8.c**

This simple example configures the ports for 8-bit output and executes writes for all output ports on all installed TPMC680 boards.

Program flow:

- open devices
- configure all ports for 8-bit output
- periodically write values to all ports
- close devices

### **4.1.3 tpmc680\_in\_16.c**

This simple example configures the port 0 and 2 for 16-bit input and receives data on the 16-bit input ports.

Program flow:

- open devices
- configure ports 0 and 2 for 16-bit input
- periodically execute receive on all ports
- close devices

### **4.1.4 tpmc680\_out\_16.c**

This simple example configures the port 0 and 2 for 16-bit output and receives data on the 16-bit output ports.

Program flow:

- open devices
- configure ports 0 and 2 for 16-bit output
- periodically execute transmits data on ports
- close devices



### 4.1.5 tpmc680\_in\_32.c

This simple example configures the port 0 for 32-bit input and receives data on the 32-bit input ports.

Program flow:

- open devices
- configure port 0 for 32-bit input
- periodically execute receive on all ports
- close devices

### 4.1.6 tpmc680\_out\_32.c

This simple example configures the port 0 for 32-bit output and transmits data on the 32-bit output ports.

Program flow:

- open devices
- configure port 0 for 32-bit output
- periodically transmits data on 32-bit port
- close devices

### 4.1.7 tpmc680\_in\_64.c

This simple example configures the TPMC680 for 64-bit input and executes reads for the 64-bit port on all installed TPMC680 boards.

Program flow:

- open devices
- configure TPMC680 for 64-bit input
- periodically execute read on 64-bit port
- close devices

### 4.1.8 tpmc680\_out\_64.c

This simple example configures the TPMC680 for 64-bit output and executes writes for the 64-bit port on all installed TPMC680 boards.

Program flow:

- open devices
- configure TPMC680 for 64-bit output
- periodically write new data on 64-bit port
- close devices

## 4.1.9 tpmc680\_event.c

This simple example configures the TPMC680 for 64-bit input and executes some event wait functions.

Program flow:

- open devices
- configure TPMC680 for 64-bit input
- wait for event (high transition) on port 0 line 0
- wait for event (low transition) on port 1 line 1
- wait for event (any transition) on port 7 line 7
- close devices