The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC680-SW-42

VxWorks Device Driver

8 x 8 Bit Digital I/O

Version 5.0.x

User Manual

Issue 5.0.0

December 2017

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

TPMC680-SW-42

VxWorks Device Driver

8 x 8 Bit Digital I/O

Supported Modules:
TPMC680-10

Issue	Description	Date
1.0	First Issue	November 16, 2002
1.1	Read() parameter description changed	December 11, 2002
1.2	Flags “empty / not full” corrected	December 13, 2002
2.0.0	New driver start and device creation functions, new file list	May 24, 2006
3.0.0	Support for VxBus and API description added, general revision read() and write() functions replaced by ioctl() functions	February 3, 2010
3.0.1	Legacy vs. VxBus Driver modified	March 26, 2010
4.0.0	New API, Manual and Driver revised	June 29, 2012
5.0.0	VxWorks 7 support added, new installation guide	December 8, 2017

Table of Contents

1 Introduction

The TPMC680-SW-42 VxWorks device driver software allows the operation of the supported PMCs conforming to the VxWorks I/O system specification.

The TPMC680-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new VxBus-enabled (GEN1 and GEN2) driver model. The VxBus-enabled driver is recommended for new developments with later VxWorks 6.x and 7.x releases and mandatory for VxWorks 64-bit and SMP systems.

Both drivers, legacy and VxBus, share the same application programming interface (API).

The TPMC680-SW-42 device driver supports the following features:

- direct reading for input ports (8 bit / synchronous mode)
- direct writing for output ports (8 bit / synchronous mode)
- 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-82 device driver 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 TPMC680 devices it is recommended to read the manuals listed below.

TPMC680 User Manual
TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide

2 API Documentation

2.1 General Functions

2.1.1 tpmc680Open

NAME

tpmc680Open – opens a device.

SYNOPSIS

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

DESCRIPTION

Before I/O can be performed to a device, a device 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 the specified device
*/
hdl = tpmc680Open(“/tpmc680/0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails

ERROR CODES

The error codes are stored in *errno.*

The error code is a standard error code set by the I/O system.

2.1.2 tpmc680Close

NAME

tpmc680Close – Closes 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

2.2 Device Access Functions

2.2.1 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 than 8 bit is 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 (8 bit)	6 (8 bit)	5 (8 bit)	4 (8 bit)	3 (8 bit)	2 (8 bit)	1 (8 bit)	0 (8 bit)
	7 (8 bit)	6 (8 bit)	5 (8 bit)	4 (8 bit)	3 (8 bit)	2 (8 bit)	0 (16 bit / HS)	
	7 (8 bit)	6 (8 bit)	5 (8 bit)	4 (8 bit)	2 (16 bit / HS)		1 (8 bit)	0 (8 bit)
	7 (8 bit)	6 (8 bit)	5 (8 bit)	4 (8 bit)	2 (16 bit / HS)		0 (16 bit / HS)	
	7 (8 bit)	6 (8 bit)	5 (8 bit)	4 (8 bit)	0 (32 bit / HS)			
	0 (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 used 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. **If this mode is selected for any port the handshake port 4 will be configured as an 8-bit input port.**
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. **If this mode is selected the handshake port 4 will be configured as an 8-bit input port.**
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 handshake mode will be used.
TPMC680_MODE_HSFLAG_PULSED	The pulsed 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 a handshake mode 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_INVAL	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.

Other returned error codes are system error conditions.

2.2.2 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.

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.

Other returned error codes are system error conditions.

2.2.3 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 written. 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.

Other returned error codes are system error conditions.

2.2.4 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_INVAL	A specified pointer is NULL.
TPMC680_ERR_ACCESS	Access not allowed with current port configuration.

Other returned error codes are system error conditions.

2.2.5 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 is stored in bits 0...7, Port 1 in bits 8…15, and so on.

portVal32_63

This argument specifies the new output value of the ports 4...7. Port 4 is stored in bits 0...7, Port 5 in 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.

Other returned error codes are system error conditions.

2.2.6 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 of 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.

Other returned error codes are system error conditions.

2.2.7 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 places the data into a FIFO and starts transmission. It will not wait until data is physically transmitted.

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 of 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 sent 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.

Other returned error codes are system error conditions.

2.2.8 tpmc680Receive32

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 of 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 to.

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.

Other returned error codes are system error conditions.

2.2.9 tpmc680Send32

NAME

tpmc680Send32 – Send data on 32-bit port

SYNOPSIS

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

DESCRIPTION

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

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.

portNo

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

bufSize

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

pBuf

This pointer points to the output buffer containing the data ready to be sent.

pSentData

This pointer points to an unsigned int value where the number of successfully sent 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,
                             0,
                             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.

Other returned error codes are system error conditions.

2.2.10 tpmc680WaitForEvent

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 ports 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. If the function shall never timeout a value of TPMC680_WAIT_FOREVER must be specified.

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

Other returned error codes are system error conditions.

3 Driver Configuration

3.1 Configuration of FIFO Depth

The depth of the FIFOs can be configured with the define *TPMC680_FIFO_SIZE* in tpmc680def.h. The value defines the number of values that can be stored in each of the FIFOs. Changing this value will change the size of the used system memory for each devices.

After changing the definition of *TPMC680_FIFO_SIZE* the driver must be rebuilt to make the changes take effect.

4 Debugging and Diagnostic

The TPMC680 device driver provides a function and debug statements to display versatile information of the driver installation and status on the debugging console.

If the VxBus driver is used, the TPMC680 show routine is included in the driver by default and can be called from the VxWorks shell. If this function is not needed or program space is rare the function can be removed from the code by un-defining the macro INCLUDE_TPMC680_SHOW in tpmc680drv.c

The tpmc680Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TPMC680 modules.

If TPMC680 modules were probed but no devices were created it may be helpful to enable debugging code inside the driver code by defining the macro TPMC680_DEBUG in tpmc680drv.c.

In contrast to VxBus TPMC680 devices, legacy TPMC680 devices must be created "manually". This will be done with the first call to the tpmc680Open API function.

```
-> tpmc680Show
Probed Modules:
    [0] : Bus=0, Dev=16, DevId=0x02a8, VenId=0x1498, Init=OK, vxDev=0x2854d8
    [1] : Bus=0, Dev=17, DevId=0x02a8, VenId=0x1498, Init=OK, vxDev=0x2855d8

Associated Devices:
    [0] : /tpmc680/0
    [1] : /tpmc680/1
value = 1 = 0x1
```