The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC680-SW-65

Windows Device Driver

64 Digital Inputs/Outputs

Version 2.0.x

User Manual

Issue 2.0.1

March 2018

powerBridge Computer

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

www.powerbridge.de
info@powerbridge.de

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

TPMC680-SW-65

Windows Device Driver

64 Digital Inputs/Outputs

Supported Modules:
TPMC680-10

Issue	Description	Date
1.0	First Issue	January 16, 2003
1.0.1	File list changed, Installation chapter reviewed	April 18, 2005
1.0.2	Title corrected	June 3, 2005
1.0.3	New Address of TEWS LLC, File list changed	January 11, 2007
1.0.4	General Revision, Return value of close() corrected	May 15, 2007
1.0.5	Files moved to subdirectory	June 23, 2008
2.0.0	Windows 7 support added, interface naming definitions modified	February 25, 2011
2.0.0	General description of Installation and Windows support	March 6, 2018

Table of Contents

1 INTRODUCTION 4

2 INSTALLATION 5

2.1 Software Installation 5

2.1.1 Windows XP 5

2.1.2 Windows 7 and newer 6

2.2 Confirming Windows Driver Installation 6

3 DRIVER CONFIGURATION 7

3.1 FIFO Configuration 7

4 DEVICE DRIVER PROGRAMMING 8

4.1 TPMC680 Files and I/O Functions 8

4.1.1 Opening a Device 8

4.1.2 Closing a Device 10

4.1.3 TPMC680 Device I/O Control Functions 11

4.1.3.1 IOCTL_TPMC680_READ8 13

4.1.3.2 IOCTL_TPMC680_READ16 15

4.1.3.3 IOCTL_TPMC680_READ32 17

4.1.3.4 IOCTL_TPMC680_READ64 19

4.1.3.5 IOCTL_TPMC680_WRITE8 21

4.1.3.6 IOCTL_TPMC680_WRITE16 23

4.1.3.7 IOCTL_TPMC680_WRITE32 26

4.1.3.8 IOCTL_TPMC680_WRITE64 28

4.1.3.9 IOCTL_TPMC680_SETMODE 30

4.1.3.10 IOCTL_TPMC680_EVENTWAIT 34

1 Introduction

The TPMC680-SW-65 Windows device driver is a kernel mode driver which allows the operation of the supported hardware module on an Intel or Intel-compatible Windows operating system.

The standard file and device (I/O) functions (CreateFile, CloseHandle, and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

The TPMC680-SW-65 device driver supports the following features:

- direct reading and writing for output ports (8 bit / synchronous mode)
- direct reading for input 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
- waiting for an input event (8 bit / synchronous mode)

The TPMC680-SW-65 device driver supports the modules listed below:

TPMC680-10	64 Digital Inputs/Outputs	(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

2 Installation

Following files are located in directory TPMC680-SW-65 on the distribution media:

i386\	Directory containing driver files for 32bit Windows versions
amd64\	Directory containing driver files for 64bit Windows versions
installer_32bit.exe	Installation tool for 32bit systems (Windows XP or later)
installer_64bit.exe	Installation tool for 64bit systems (Windows XP or later)
tpmc680.inf	Windows installation script
tpmc680.h	Header file with IOCTL codes and structure definitions
example\tpmc680exa.c	Example application
TPMC680-SW-65-2.0.1.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

For installation the files have to be copied to the desired target directory.

2.1 Software Installation

2.1.1 Windows XP

This section describes how to install the TPMC680 Device Driver on a Windows XP operating system.

After installing the TPMC680 card(s) and boot-up your system, Windows XP setup will show a "***New hardware found***" dialog box.

1. The "***Upgrade Device Driver Wizard***" dialog box will appear on your screen.
   Click "***Next***" button to continue.
2. In the following dialog box, choose "***Search for a suitable driver for my device***".
   Click "***Next***" button to continue.
3. Insert the TPMC680 driver media; select "***Disk Drive***" in the dialog box.
   Click "***Next***" button to continue.
4. Now the driver wizard should find a suitable device driver on the media.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Now copy all needed files (tpmc680.h) to the desired target directories.

After successful installation the TPMC680 device driver will start immediately and creates devices (TPMC680_1, TPMC680_2 ...) for all recognized TPMC680 modules.

2.1.2 Windows 7 and newer

This section describes how to install the TPMC680-SW-65 Device Driver on a Windows 7 (32bit or 64bit) operating system.

Depending on the operating system type, execute the installer binaries for either 32bit or 64bit systems. This will install all required driver files using an installation wizard.

Copy needed files (tpmc680.h) to desired target directory.

After successful installation a device is created for each module found (TPMC680_1, TPMC680_2 ...).

2.2 Confirming Windows Driver Installation

To confirm that the driver has been properly loaded, perform the following steps:

1. Open the Windows Device Manager:
    a. For Windows XP, open the "***Control Panel***" from "***My Computer***" and click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
    b. For Windows 7, open the "***Control Panel***" from "***My Computer***" and then click the "***Device Manager***" entry.
2. Click the "**+**" in front of "***Embedded I/O***".
   The driver "***TEWS TECHNOLOGIES – TPMC680 (64 digital I/O) (TPMC680-10)***" should appear for each installed device.

3 Driver Configuration

3.1 FIFO Configuration

After Installation of the TPMC680 Device Driver the FIFO size is set to its default value.

The default value is 100.

If the default value is not suitable the configuration can be changed by modifying the registry, for instance with regedit.

To change the size of the FIFO the following value must be modified.

```
HKLM\System\CurrentControlSet\Services\TPMC680\Parameters\FifoSize
```

The size value must be greater than 2

4 Device Driver Programming

The TPMC680-SW-65 Windows WDM device driver is a kernel mode device driver.

The standard file and device (I/O) functions (CreateFile, CloseHandle, and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

All of these standard Win32 functions are described in detail in the Windows Platform SDK Documentation (Windows base services / Hardware / Device Input and Output).

For details refer to the Win32 Programmers Reference of your used programming tools (C++, Visual Basic etc.)

4.1 TPMC680 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TPMC680 device driver. Only the required parameters are described in detail.

4.1.1 Opening a Device

Before you can perform any I/O the *TPMC680* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TPMC680* device.

```
HANDLE CreateFile(
    LPCTSTR                 lpFileName,
    DWORD                   dwDesiredAccess,
    DWORD                   dwShareMode,
    LPSECURITY_ATTRIBUTES   lpSecurityAttributes,
    DWORD                   dwCreationDistribution,
    DWORD                   dwFlagsAndAttributes,
    HANDLE                  hTemplateFile
);
```

Parameters

lpFileName

This parameter points to a null-terminated string, which specifies the name of the TPMC680 to open. The *lpFileName* string should be of the form **\\.\TPMC680_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TPMC680_1, the second \\.\TPMC680_2 and so on.

dwDesiredAccess

This parameter specifies the type of access to the TPMC680.
For the TPMC680 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

dwShareMode

Set of bit flags that specify how the object can be shared. Set to 0.

lpSecurityAttributes

This argument is a pointer to a security structure. Set to NULL for TPMC680 devices.

dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TPMC680 devices must be always opened **OPEN_EXISTING**.

dwFlagsAndAttributes

Specifies the file attributes and flags for the file. This value must be set to 0 (no overlapped I/O).

hTemplateFile

This value must be NULL for TPMC680 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TPMC680 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TPMC680_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,               // no security attrs
     OPEN_EXISTING,      // TPMC680 device always open existing
     0,                  // no overlapped I/O
     NULL
);

if (hDevice == INVALID_HANDLE_VALUE) {
     ErrorHandler( "Could not open device" ); // process error
}
```

See Also

CloseHandle(), Win32 documentation CreateFile()

4.1.2 Closing a Device

The **CloseHandle** function closes an open TPMC680 handle.

BOOL CloseHandle(
HANDLE *hDevice;*
);

Parameters

BOOLEAN hDevice

Identifies an open TPMC680 handle.

Return Value

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE). To get extended error information, call ***GetLastError*.**

Example

```
HANDLE    hDevice;

if( !CloseHandle( hDevice ) ) {
     ErrorHandler( "Could not close device" ); // process error
}
```

See Also

CreateFile (), Win32 documentation CloseHandle ()

4.1.3 TPMC680 Device I/O Control Functions

The **DeviceIoControl** function sends a control code directly to a specified device driver, causing the corresponding device to perform the specified operation.

```
BOOL DeviceIoControl(
    HANDLE          hDevice,
    DWORD           dwIoControlCode,
    LPVOID          lpInBuffer,
    DWORD           nInBufferSize,
    LPVOID          lpOutBuffer,
    DWORD           nOutBufferSize,
    LPDWORD         lpBytesReturned,
    LPOVERLAPPED    lpOverlapped
);
```

Parameters

hDevice

Handle to the TPMC680 that is to perform the operation.

dwIoControlCode

This parameter specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in tpmc680.h:

Value	Meaning
IOCTL_TPMC680_READ8	Read a port value from an 8 bit port
IOCTL_TPMC680_READ16	Read a buffered value from a 16 bit input port
IOCTL_TPMC680_READ32	Read a buffered value from a 32 bit input port
IOCTL_TPMC680_READ64	Read a port value from a 64 bit port
IOCTL_TPMC680_WRITE8	Write a port value to an 8 bit output port
IOCTL_TPMC680_WRITE16	Write buffered to a 16 bit buffered output port
IOCTL_TPMC680_WRITE32	Write buffered to a 32 bit buffered output port
IOCTL_TPMC680_WRITE64	Write a port value to a 64 bit output port
IOCTL_TPMC680_SETMODE	Change port mode and direction
IOCTL_TPMC680_EVENTWAIT	Wait for a specified input event

See behind for more detailed information on each control code.

lpInBuffer

Pointer to a buffer that contains the data required to perform the operation.

nInBufferSize

This argument specifies the size, in bytes, of the buffer pointed to by *lpInBuffer*.

lpOutBuffer

Pointer to a buffer that receives the operation's output data.

nOutBufferSize

This argument specifies the size, in bytes, of the buffer pointed to by *lpOutBuffer*.

lpBytesReturned

Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by *lpOutBuffer*. A valid pointer is required.

lpOverlapped

This argument is a pointer to an *Overlapped* structure. This value must be set to NULL (no overlapped I/O).

To use these TPMC680 specific control codes the header file tpmc680.h must be included.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call ***GetLastError***.

Note. The TPMC680 device driver returns always standard Win32 error codes on failure, please refer to the Windows Platform SDK Documentation for a detailed description of returned error codes.

See Also

Win32 documentation DeviceIoControl ()

4.1.3.1 IOCTL_TPMC680_READ8

This TPMC680 control function reads an 8 bit value directly from the specified port. A pointer to the port number (*ULONG*) is passed by the parameter *lpInBuffer*. The pointer to the return buffer is passed by the parameter *lpOutBuffer* to the driver.

After successful execution the port value (*UCHAR*) is returned in the specified return buffer (*lpOutBuffer)*.

Example

```
#include "tpmc680.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
ULONG     portNo;
UCHAR     val8;

portNo = 5;                 // read from port 5

success = DeviceIoControl (
     hDevice,               // TPMC680 handle
     IOCTL_TPMC680_READ8,        // control code
     &portNo,               // buffer with control information
     sizeof(portNo),
     &val8,                 // buffer which receives the port value
     sizeof(UCHAR),
     &NumBytes,             // number of bytes transferred
     NULL
);

if( success ) {
     // Process data
     printf("INPUT: %02Xh\n", val8);
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.2 IOCTL_TPMC680_READ16

This TPMC680 control function reads 16 bit values from the specified buffered input port. A pointer to the port number (*ULONG*) is passed by the parameter *lpInBuffer*. The pointer to the return buffer is passed by the parameter *lpOutBuffer* to the driver.

After successful execution a filled array of 16 bit values (*USHORT*) is returned in the specified return buffer (*lpOutBuffer)*.

The number of 16 bit data read with this function by maximum is limited with the buffer size. If there are less data values stored in the FIFO, only the already received data values will be returned and the *NumBytes* specifies the size of valid data.

Example

```
#include "tpmc680.h"

#define BUFMAX 10

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
ULONG     portNo;
USHORT    val16[BUFMAX];
int       i;

//
// read 10 16bit words from port 2
//
portNo = 2;
success = DeviceIoControl (
     hDevice,                // TPMC680 handle
     IOCTL_TPMC680_READ16,   // control code
     &portNo,                // buffer with control information
     sizeof(portNo),
     &val16[0],              // buffer which receives the port value
     sizeof(USHORT) * BUFMAX,
     &NumBytes,              // number of bytes transferred
     NULL
);

…
```

```
…

if( success ) {
     // Process data
     for (i = 0; i < BUFMAX; i++) {
          printf("(%d): %04Xh\n", i, val16[i]);
     }
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.3 IOCTL_TPMC680_READ32

This TPMC680 control function reads 32 bit values from the specified buffered input port. A pointer to the port number (*ULONG*) is passed by the parameter *lpInBuffer*. The pointer to the return buffer is passed by the parameter *lpOutBuffer* to the driver.

After successful execution a filled array of 32 bit values (*ULONG*) is returned in the specified return buffer (*lpOutBuffer)*.

The number of 32 bit data read with this function by maximum is limited with the buffer size. If there are less data values stored in the FIFO, only the already received data values will be returned and the *NumBytes* specifies the size of valid data.

Example

```
#include "tpmc680.h"

#define BUFMAX 10

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
ULONG     portNo;
ULONG     val32[BUFMAX];

//
// read 10 32bit words from port 0
//
portNo = 0;
success = DeviceIoControl (
     hDevice,               // TPMC680 handle
     IOCTL_TPMC680_READ32,  // control code
     &portNo,               // buffer with control information
     sizeof(portNo),
     &val32[0],             // buffer which receives the port value
     sizeof(ULONG) * BUFMAX,
     &NumBytes,             // number of bytes transferred
     NULL
);
…
```

```
…
if( success ) {
     // Process data
     for (i = 0; i < BUFMAX; i++) {
          printf("(%d): %08lXh\n", i, val32[i]);
     }
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.4 IOCTL_TPMC680_READ64

This TPMC680 control function reads the current 64 bit value from the specified port. A pointer to the port number (*ULONG*) is passed by the parameter *lpInBuffer*. The pointer to the return buffer is passed by the parameter *lpOutBuffer* to the driver.

After successful execution a filled structure with 64 bit value (2**ULONG*) is returned in the specified return buffer (*lpOutBuffer)*.

Example

```
#include "tpmc680.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
ULONG     portNo;
ULONG     val64[2];

//
// read 1 64bit word from port 0
//
portNo = 0;
success = DeviceIoControl (
     hDevice,                 // TPMC680 handle
     IOCTL_TPMC680_READ64,    // control code
     &portNo,                 // buffer with control information
     sizeof(portNo),
     &val64[0],               // buffer which receives the port value
     sizeof(ULONG) * 2,
     &NumBytes,               // number of bytes transferred
     NULL
);

if( success ) {
     // Process data
     printf("INPUT: %08lX %08lX h\n", val64[0], val64[1]);
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.5 IOCTL_TPMC680_WRITE8

This TPMC680 control function writes an 8 bit value directly to the specified output port. A pointer to the write buffer (*TPMC680_WRITE_8BIT_BUF*) is passed by the parameter *lpInBuffer* to the driver.

The *lpOutBuffer* is not used and should be a *NULL* pointer.

```
typedef struct {
    ULONG    portNo;    // Port number to handle
    UCHAR    data;      // 8 bit data
} TPMC680_WRITE_8BIT_BUF, *PTPMC680_WRITE_8BIT_BUF;
```

Members

portNo

This member specifies the port that shall be changed. Valid values are 0 up to 7.

data

This argument specifies the new output value.

Example

```
#include "tpmc680.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TPMC680_WRITE_8BIT_BUF  wr8Buf;

wr8Buf.portNo = 3;          // write to port 3
wr8Buf.data   = 0x55;       // new output value

success = DeviceIoControl (
     hDevice,               // TPMC680 handle
     IOCTL_TPMC680_WRITE8,  // control code
     &wr8Buf,               // buffer with control information
     sizeof(wr8Buf),
     NULL,                  // buffer which receives the port value
     0,
     &NumBytes,             // number of bytes transferred
     NULL
);

…
```

```
…

if( success ) {
     // Write OK
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.6 IOCTL_TPMC680_WRITE16

This TPMC680 control function writes 16 bit values buffered to the output. A pointer to the write buffer (with head *TPMC680_WRITE_16BIT_BUF*) is passed by the parameter *lpInBuffer* to the driver. The buffer size depends on the number of data that shall be transferred. For calculating the memory amount needed for the specified number of data can be calculated with the *TPMC680_BUFSIZE16(<number of data values>)* macro.

The *lpOutBuffer* is not used and should be a *NULL* pointer.

The number of send bytes will be returned in NumBytes.

```
typedef struct {
    ULONG    portNo;    // Port number to handle
    ULONG    numData;   // Number of Data values
    USHORT   data[1];   // 16 bit data buffer
} TPMC680_WRITE_16BIT_BUF, *PTPMC680_WRITE_16BIT_BUF;
```

Members

portNo

This member specifies the port that shall be changed. Valid values are 0 and 2.

numData

This argument specifies the number of data values (16 bit) following.

data[1]

This array prototype specifies the beginning of the output values.

Example

```
#include "tpmc680.h"

HANDLE                    hDevice;
BOOLEAN                   success;
ULONG                     NumBytes;
TPMC680_WRITE_16BIT_BUF   *pwr16Buf;
int                       bufSize;

...
```

```
…

// Get buffer
bufSize   = TPMC680_BUFSIZE16(8); // 8 data values shall be written
pwr16Buf  = (TPMC680_WRITE_16BIT_BUF*)malloc(bufSize);

pwr16Buf->portNo   = 2;          // write to port 2
pwr16Buf->numData  = 8;          // 8 data values shall be written
pwr16Buf->data[0]  = 0x1111;
pwr16Buf->data[1]  = 0x2222;
pwr16Buf->data[2]  = 0x3333;
pwr16Buf->data[3]  = 0x4444;
pwr16Buf->data[4]  = 0x5555;
pwr16Buf->data[5]  = 0x6666;
pwr16Buf->data[6]  = 0x7777;
pwr16Buf->data[7]  = 0x8888;

success = DeviceIoControl (
     hDevice,                // TPMC680 handle
     IOCTL_TPMC680_WRITE16,  // control code
     pwr16Buf,               // buffer with control information
     bufSize,
     NULL,
     0,
     &NumBytes,              // number of bytes transferred
     NULL
);

if( success ) {
     // Write OK
}
else {
     // Process DeviceIoControl() error
}

free (pwr16Buf);
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.7 IOCTL_TPMC680_WRITE32

This TPMC680 control function writes 32 bit values buffered to the output. A pointer to the write buffer (with head *TPMC680_WRITE_32BIT_BUF*) is passed by the parameter *lpInBuffer* to the driver. The buffer size depends on the number of data that shall be transferred. For calculating the memory amount needed for the specified number of data can be calculated with the *TPMC680_BUFSIZE32(<number of data values>)* macro. The *lpOutBuffer* is not used and should be a *NULL* pointer.

The number of send bytes will be returned in NumBytes.

```
typedef struct {
    ULONG    portNo;        // Port number to handle
    ULONG    numData;       // Number of Data values
    ULONG    data[1];       // 32 bit data buffer
} TPMC680_WRITE_32BIT_BUF, *PTPMC680_WRITE_32BIT_BUF;
```

Members

portNo

This member specifies the port that shall be changed. The only valid value is 0.

numData

This argument specifies the number of data values (32 bit) following.

data[1]

This array prototype specifies the beginning of the output values.

Example

```
#include "tpmc680.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TPMC680_WRITE_32BIT_BUF *pwr32Buf;
int                     bufSize;

// Get buffer
bufSize   = TPMC680_BUFSIZE32(6); // 6 data values shall be written
pwr32Buf  = (TPMC680_WRITE_32BIT_BUF*)malloc(bufSize);

...
```

```
…

pwr32Buf->portNo   = 2;          // write to port 2
pwr32Buf->numData  = 6;          // 6 data values shall be written
pwr32Buf->data[0]  = 0x11223344;
pwr32Buf->data[1]  = 0x22334455;
pwr32Buf->data[2]  = 0x33445566;
pwr32Buf->data[3]  = 0x44556677;
pwr32Buf->data[4]  = 0x55667788;
pwr32Buf->data[5]  = 0x66778899;

success = DeviceIoControl (
     hDevice,                // TPMC680 handle
     IOCTL_TPMC680_WRITE32,  // control code
     pwr32Buf,               // buffer with control information
     bufSize,
     NULL,
     0,
     &NumBytes,              // number of bytes transferred
     NULL
);

if( success ) {
     // Write OK
}
else {
     // Process DeviceIoControl() error
}

free (pwr32Buf);
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.8 IOCTL_TPMC680_WRITE64

This TPMC680 control function writes a 64 bit value directly to the specified output port. A pointer to the write buffer (*TPMC680_WRITE_64BIT_BUF*) is passed by the parameter *lpInBuffer* to the driver.

The *lpOutBuffer* is not used and should be a *NULL* pointer.

```
typedef struct {
    ULONG    portNo;    // Port number to handle
    ULONG    data[2];   // 64 bit data
} TPMC680_WRITE_64BIT_BUF, *PTPMC680_WRITE_64BIT_BUF;
```

Members

portNo

This member specifies the port that shall be changed. The only valid value is 0.

data

This array specifies the new output value. Index 0 specifies the output for ports 7..4 and index 1 specifies the value for ports 3..0.

Example

```
#include "tpmc680.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TPMC680_WRITE_64BIT_BUF wr64Buf;

wr64Buf.portNo     = 0;          // write to port 0
wr64Buf.data[0]    = 0x77665544; // new output value port 7/6/5/4
wr64Buf.data[1]    = 0x33221100; // new output value port 3/2/1/0

success = DeviceIoControl (
     hDevice,               // TPMC680 handle
     IOCTL_TPMC680_WRITE64, // control code
     &wr64Buf,              // buffer with control information
     sizeof(wr64Buf),
     NULL,
     0,
     &NumBytes,             // number of bytes transferred
     NULL
);
…
```

```
…

if( success ) {
     // Write OK
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode

All other returned error codes are system error conditions.

4.1.3.9 IOCTL_TPMC680_SETMODE

This TPMC680 control function configures the port size and direction. A pointer to the configuration buffer (*TPMC680_MODE_BUF*) is passed by the parameter *lpInBuffer* to the driver.

The *lpOutBuffer* is not used and should be a *NULL* pointer.

```
typedef struct {
    ULONG    portNo;     // Port number to handle
    ULONG    Size;       // Port size
    ULONG    Direction;  // Port direction
    ULONG    HSMode;     // Handshake Output Mode
    ULONG    HSFifo;     // Handshake Output Fifo Mode
} TPMC680_MODE_BUF, *PTPMC680_MODE_BUF;
```

Members

portNo

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

Size

This argument specifies the port size. 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.

Direction

This member specifies direction of the port. All connected ports will get 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.

HSMode

This value specifies the handshake mode and is only valid if the port shall be configured in 16 or 32 bit handshake mode (*TPMC680_MODE_SIZE_16BIT, TPMC680_MODE_SIZE_32BIT*). Using an output handshake, will change the direction of port 4 to input and 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.

HSFifo

This value specifies the handshake event depending on the handshake FIFO fill level. This value is only used if an output handshake is configured. The 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.

Changing a port size from big to small will also change the mode of the previously connected ports. The ports will be set into 8 bit mode and they will keep their direction.

Example

```
#include "tpmc680.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TPMC680_MODE_64BIT_BUF  modeBuf;

//
// setup port 2 for 16 bit Handshake mode
// port direction is output, HS output shall be pulsed mode
// HS event on FIFO empty
//
modeBuf.portNo     = 2;
modeBuf.Size       = TPMC680_MODE_SIZE_16BIT;
modeBuf.Direction  = TPMC680_MODE_DIR_OUTPUT;
modeBuf.HSMode     = TPMC680_MODE_HSFLAG_PULSED;
modeBuf.HSFifo     = TPMC680_MODE_HSFIFOEV_EMPTY;

// This setting will affect port 2, 3 and the HS ports 4 and 5

success = DeviceIoControl (
     hDevice,                // TPMC680 handle
     IOCTL_TPMC680_SETMODE,  // control code
     &modeBuf,               // buffer with control information
     sizeof(modeBuf),
     NULL,
     0,
     &NumBytes,              // number of bytes transferred
     NULL
);

if( success ) {
     // Setup OK
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port number, Size, Handshake or FIFO mode is specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode. Another port is controls this port.

All other returned error codes are system error conditions.

4.1.3.10 IOCTL_TPMC680_EVENTWAIT

This TPMC680 control function waits until a specified input event occurs. A pointer to the event buffer (*TPMC680_EVENT_BUF*) is passed by the parameter *lpInBuffer* to the driver.

The *lpOutBuffer* is not used and should be a *NULL* pointer.

```
typedef struct {
    ULONG    portNo;        // Port number to handle
    ULONG    lineNo;        // Input Line, event shall occur on
    ULONG    transition;    // Specify transition
    ULONG    timeout;       // timeout in seconds
} TPMC680_EVENT_BUF, *PTPMC680_EVENT_BUF;
```

Members

portNo

This member specifies the port to wait for. Valid values are between 0 and 7.

lineNo

This member specified the line to wait for. Valid values are between 0 and 7.

transition

This member specifies the 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 seconds. If the specified event does not occur in the specified time, the function will return with an error code. Specify a negative value to wait indefinitely.

This function is only supported for 8 bit and 64 bit ports. Other configurations will return an error code.

Example

```
#include "tpmc680.h"

HANDLE                    hDevice;
BOOLEAN                   success;
ULONG                     NumBytes;
TPMC680_EVENT_64BIT_BUF   evBuf;

//
// Wait for an event on port 4, line 6
// Selected event: high-to-low transition (falling edge)
// timeout after 15 seconds
//
evBuf.portNo       = 4;
evBuf.lineNo       = 6;
evBuf.transition   = TPMC680_IO_EDGE_LO;
evBuf.timeout      = 15;

success = DeviceIoControl (
     hDevice,                // TPMC680 handle
     IOCTL_TPMC680_EVENTWAIT,// control code
     &evBuf,                 // buffer with control information
     sizeof(evBuf),
     NULL,
     0,
     &NumBytes,              // number of bytes transferred
     NULL
);

if( success ) {
     // Event occurred
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the message buffer is too small.
ERROR_INVALID_PARAMETER	Invalid port, line number or event is specified.
ERROR_NOACCESS	This function is not allowed for this port in the configured mode.
ERROR_IRQ_BUSY	Another process is already waiting for this event.
ERROR_SEM_TIMEOUT	The specified time has expired without the event occurred.
ERROR_OPERATION_ABORTED	The event wait operation has been cancelled.

All other returned error codes are system error conditions.