The Embedded I/O Company

TPMC700-SW-65

Windows WDM Device Driver

32(16) Digital Output PMC

Version 1.0.x

User Manual

Issue 1.0.1

January 2010

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

TPMC700-SW-65

Windows WDM Device Driver

32(16) Digital Output PMC

Supported Modules:
TPMC700

Issue	Description	Date
1.0	First Issue	August 28, 2003
1.0.1	File list changed, general revision	January 25, 2010

Table of Contents

1 Introduction

The TPMC700-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TPMC700 on an Intel or Intel-compatible x86 Windows 2000, Windows XP, Windows XP embedded 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 TPMC700-SW-65 device driver supports the following features:

- writing a new output value
- start and stop the output watchdog
- acknowledge watchdog errors

The TPMC700-SW-65 device driver supports the modules listed below:

TPMC700	32/16 Digital Outputs	(PMC)

To get more information about the features and use of TPMC700 devices it is recommended to read the manuals listed below.

TPMC700 User Manual

TPMC700 Engineering Manual

2 Installation

Following files are located on the distribution media:

Directory path ‘TPMC700-SW-65’:

tpmc700.sys	Windows driver binary
tpmc700.h	Header-file with IOCTL code definitions
tpmc700.inf	Windows installation script
example/tpmc700exa.c	Microsoft Visual C example application
EmbeddedIoDeviceClass.dll	Windows WDM device class library
TPMC700-SW-65-1.0.1.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TPMC700 Device Driver on a Windows 2000 / XP operating system.

After installing the TPMC700 card(s) and boot-up your system, Windows 2000 / 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 TPMC700 driver disk or CD-Rom into drive; select "***Disk Drive***" or “***CD_Rom Drive***” in the dialog box.
   Click "***Next***" button to continue.
4. Now the driver wizard should find a suitable device driver on the diskette or CD-Rom.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.

After successful installation the TPMC700 device driver will start immediately and create devices (TPMC700_1, TPMC700_2 ...) for all recognized TPMC700 modules.

2.1.2 Confirming Windows 2000 / XP Installation

To confirm the driver has been properly loaded in Windows 2000 / XP, perform the following steps:

1. From Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***".
2. Click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
3. Click the "+" in front of " Embedded I/O ".
   The driver "TEWS TECHNOLOGIES - TPMC700 (32/16 Digital Output)" should appear.

3 TPMC700 Device Driver Programming

The TPMC700-SW-65 Windows WDM device driver is a kernel mode device driver.

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

3.1 TPMC700 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TPMC700 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TPMC700 Device

Before you can perform any I/O the *TPMC700* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TPMC700* device.

```
HANDLE CreateFile(
        LPCTSTR lpFileName,                          // pointer to name of the file
        DWORD dwDesiredAccess,                       // access (read-write) mode
        DWORD dwShareMode,                           // share mode
        LPSECURITY_ATTRIBUTES lpSecurityAttributes,  // pointer to security attributes
        DWORD dwCreationDistribution,                // how to create
        DWORD dwFlagsAndAttributes,                  // file attributes
        HANDLE hTemplateFile                         // handle to file with attributes to copy
);
```

lpFileName

Points to a null-terminated string, that specifies the name of the TPMC700 to open.
The *lpFileName* string should be of the form **\\.\TPMC700_*x*** to open the device *x.* The ending x is a one-based number. The first device found by the driver is \\.\TPMC700_1, the second \\.\TPMC700_2 and so on.

dwDesiredAccess

Specifies the type of the access to the TPMC700.
For the TPMC700 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

dwShareMode

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

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TPMC700 devices.

dwCreationDistribution

Specifies the action to take on files that exist, and which action to take when files do not exist. TPMC700 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 TPMC700 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TPMC700 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TPMC700_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TPMC700 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()

3.1.2 Closing a TPMC700 Device

The **CloseHandle** function closes an open TPMC700 handle.

```
BOOL CloseHandle(
    HANDLE hDevice;                // handle to a TPMC700 device to close
);
```

hDevice

Identifies an open TPMC700 handle.

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*****.**

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TPMC700_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                // no security attrs
     OPEN_EXISTING,       // TPMC700 device always open existing
     0,                   // no overlapped I/O
     NULL
);

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

/* ... do some device I/O ...  */

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

See Also

CreateFile(), Win32 documentation CloseHandle()

3.1.3 TPMC700 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,                     // handle to device of interest
        DWORD dwIoControlCode,              // control code of operation to perform
        LPVOID lpInBuffer,                  // pointer to buffer to supply input data
        DWORD nInBufferSize,                // size of input buffer
        LPVOID lpOutBuffer,                 // pointer to buffer to receive output data
        DWORD nOutBufferSize,               // size of output buffer
        LPDWORD lpBytesReturned,            // pointer to variable to receive output byte count
        LPOVERLAPPED lpOverlapped           // pointer to overlapped structure for asynchronous
                                            // operation
);
```

hDevice

Handle to the TPMC700 that is to perform the operation.

dwIoControlCode

Specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in *tpmc700.h* :

Value	Meaning
IOCTL_TP700_WRITE	Write output port
IOCTL_TP700_WDENABLE	Enable output watchdog
IOCTL_TP700_WDDISABLE	Disable output watchdog
IOCTL_TP700_WDRESET	Reset output watchdog

See behind for more detailed information on each control code.

To use these TPMC700 specific control codes the header file tpmc700.h must be included in the application

lpInBuffer

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

nInBufferSize

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

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

Pointer to an *Overlapped* structure. This value must be set to NULL (no overlapped I/O).

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 TPMC700 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()

3.1.3.1 IOCTL_TP700_WRITE

This control function attempts to write the output port of the TPMC700 associated with the open device handle.

The new port value is passed in an unsigned short buffer, pointed by *lpInBuffer*, to the driver. The buffer must be always an unsigned long type independent of the TPMC700 variant. The argument *nInBufferSize* specifies the size (size of ULONG) of the write buffer.

For the TPMC700-11 and TPMC700-21 only the lower 16 bits are relevant.

Example

```
#include “tpmc700.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
ULONG     PortData;


PortData = 0x12345678;

success = DeviceIoControl (
     hDevice,          //  TPMC700 handle
     IOCTL_TP700_WRITE, //  control code
     &PortData,
     sizeof(PortData),
     NULL,
     0,
     &NumBytes,
     NULL              //  not overlapped
);
if( success ) {
     printf("\nOutput port successful written\n");
}
else {
     ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_PARAMETER	This error is returned if the size of the write buffer is too small
ERROR_IO_DEVICE	The output register is locked by a watchdog failure. Execute the control function IOCTL_TP700_WDRESET to reset the watchdog error.

3.1.3.2 IOCTL_TP700_WDENABLE

This control function enables the output watchdog function of the TPMC700 after the next write operation to the device. Please remember if the watchdog is enabled and no write access occurs within 120 ms all outputs go into the OFF state. To unlock the output register and leave the OFF state the device control function *IOCTL_TP700_WDRESET* must be executed.

No additional parameter is required for this call.

Example

```
#include “tpmc700.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;

success = DeviceIoControl (
     hDevice,                    //  TPMC700 handle
     IOCTL_TP700_WDENABLE,       //  control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                        //  not over lapped
);
if( success ) {
     printf( "\nEnable output watchdog successful\n");
}
else {
     ErrorHandler ( "Device I/O control error” );     // process error
}
```

3.1.3.3 IOCTL_TP700_WDDISABLE

This device control function disables the output watchdog function of the TPMC700 enabled by *IOCTL_TP700_WDENABLE*.

No additional parameter is required for this call.

Example

```
#include “tpmc700.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;

success = DeviceIoControl (
     hDevice,                // TPMC700 handle
     IOCTL_TP700_WDDISABLE,  // control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                    // not over lapped
);
if( success ) {
     printf( "\nDisable output watchdog successful\n");
}
else {
     ErrorHandler ( "Device I/O control error” ); // process error
}
```

3.1.3.4 IOCTL_TP700_WDRESET

This device control function resets an output watchdog error. If IOCTL_TP700_WRITE returns the error code *ERROR_IO_DEVICE* this function must be executed to unlock the output register.

No additional parameter is required for this call.

Example

```
#include “tpmc700.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;

success = DeviceIoControl (
     hDevice,                // TPMC700 handle
     IOCTL_TP700_WDRESET,    // control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                    // not over lapped
);
if( success ) {
     printf( "\nReset output watchdog successful\n");
}
else {
     ErrorHandler ( "Device I/O control error” ); // process error
}
```