The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV003-SW-65

Windows 2000/XP Device Driver

16(8) Digital I/O PMC

Version 1.0.x

User Manual

Issue 1.0.2

June 2008

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

TEWS TECHNOLOGIES LLC
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com
Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TDRV003-SW-65

Windows 2000/XP Device Driver

16(8) Digital I/O PMC

Supported Modules:
TPMC670
TPMC671

Issue	Description	Date
1.0.0	First Issue	October 12, 2005
1.0.1	Example CloseHandle() corrected, New Address TEWS LLC, file list changed	May 19, 2008
1.0.2	Files moved to subdirectory	June 23, 2008

Table of Contents

1 Introduction

The TDRV003-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the supported modules on an Intel or Intel-compatible x86 Windows 2000 or Windows XP 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 TDRV003 device driver supports the following features:

- write a new output value
- read the input port immediately without waiting for a specific input event
- read the input port after the following events occur
  - masked input bits match to the specified pattern
  - high-transition at the specified bit position
  - low-transition at the specified bit position
  - any transition (high or low) at the specified bit position
- start and stop the output watchdog
- acknowledge watchdog errors
- configure & start and stop input debouncing

The TDRV003-SW-65 supports the modules listed below:

TPMC670	16(8) Digital Input (24V) 16(8) Digital Output (24V, 0.5A) (50 pin connector)	PMC
TPMC671	16 Digital Input (24V) 16 Digital Output (24V, 0.5A) (64 pin connector)	PMC

In this document all supported modules and devices will be called TDRV003. Specials for certain devices will be advised.

To get more information about the features and use of TDRV003 devices it is recommended to read the manuals listed below.

TPMC670/TPMC671 User manual

TPMC670/TPMC671 Engineering Manual

2 Installation

Following files are located in directory TDRV003-SW-65 on the distribution media:

tdrv003.sys	Windows driver binary
tdrv003.h	Header-file with IOCTL code definitions
tdrv003.inf	Windows installation script
TDRV003-SW-65-1.0.2.pdf	This document
\example\tdrv003exa.c	Microsoft Visual C example application
Release.txt	Release Information
ChangeLog.txt	Release History

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV003 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV003 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. In Drive A, insert the TDRV003 driver disk; 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 diskette.
   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 (tdrv003.h, TDRV003-SW-65.pdf) to the desired target directories.

After successful installation the TDRV003 device driver will start immediately and creates devices (TDRV003_1, TDRV003_2 ...) for all recognized TDRV003 modules.

2.1.2 Confirming Windows 2000 / XP Installation

To confirm that 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 "***Other Devices***".
   The driver "***TEWS TECHNOLOGIES TDRV003 (Digital I/O)***" should appear.

3 Driver Configuration

3.1 Event Queue Configuration

After Installation of the TDRV003 Device Driver the number concurrent event controlled read request is limited to 10.

If the default values are not suitable the configuration can be changed by modifying the registry, for instance with regedt32.

To change the number of queue entries the following value must be modified.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tdrv003\NumReqEntries

Valid values are in range between 1..100

4 TDRV003 Device Driver Programming

The TDRV003-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 TDRV003 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TDRV003 device driver. Only the required parameters are described in detail.

4.1.1 Opening a TDRV003 Device

Before you can perform any I/O the *TDRV003* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TDRV003* device.

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

Parameters

LPCTSTR lpFileName

This parameter points to a null-terminated string, which specifies the name of the TDRV003 to open. The *lpFileName* string should be of the form **\\.\TDRV003_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TDRV003_1, the second \\.\TDRV003_2 and so on.

DWORD dwDesiredAccess

This parameter specifies the type of access to the TDRV003.
For the TDRV003 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

DWORD dwShareMode

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

LPSECURITY_ATTRIBUTES lpSecurityAttributes

This argument is a pointer to a security structure. Set to NULL for TDRV003 devices.

DWORD dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TDRV003 devices must be always opened **OPEN_EXISTING**.

DWORD dwFlagsAndAttributes

Specifies the file attributes and flags for the file. This value must be set to FILE_FLAG_OVERLAPPED for TDRV003 devices.

HANDLE hTemplateFile

This value must be NULL for TDRV003 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TDRV003 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TDRV003_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                    // no security attrs
     OPEN_EXISTING,           // TDRV003 device always open existing
     FILE_FLAG_OVERLAPPED,    // 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 TDRV003 Device

The **CloseHandle** function closes an open TDRV003 handle.

BOOL CloseHandle(
HANDLE *hDevice;*
);

Parameters

HANDLE hDevice

Identifies an open TDRV003 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;

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

See Also

CreateFile (), Win32 documentation CloseHandle ()

4.1.3 TDRV003 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
);
```

Parameters

hDevice

Handle to the TDRV003 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 *tdrv003.h*:

Value	Meaning
IOCTL_TDRV003_WRITE	Write output port
IOCTL_TDRV003_READ	Read input port immediately
IOCTL_TDRV003_READ_EVENT	Read input port after specified event occur
IOCTL_TDRV003_WDENABLE	Enable output watchdog
IOCTL_TDRV003_WDDISABLE	Disable output watchdog
IOCTL_TDRV003_WDRESET	Reset output watchdog
IOCTL_TDRV003_DEBENABLE	Enable input debounce function
IOCTL_TDRV003_DEBDISABLE	Disable input debounce function

See below for more detailed information on each control code.

To use these TDRV003 specific control codes the header file tdrv003.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 of the buffer pointed to by *lpInBuffer*.

lpOutBuffer

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

nOutBufferSize

Specifies the size of the buffer in bytes 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. Refer to the Ioctl specific manual section how this parameter must be set.

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***.

See Also

Win32 documentation DeviceIoControl()

4.1.3.1 IOCTL_TDRV003_WRITE

This control function attempts to write the output port of the TDRV003 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 short type independent of the TDRV003 variant. The argument *nInBufferSize* specifies the size (size of USHORT) of the write buffer.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant

Example

```
#include “tdrv003.h”


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


PortData = 0x1234;

success = DeviceIoControl (
     hDevice,                // TDRV003 handle
     IOCTL_TDRV003_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_TDRV003_WDRESET to reset the watchdog error.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TDRV003 Hardware User Manual

4.1.3.2 IOCTL_TDRV003_READ

This control function attempts to read the input port of the TDRV003 associated with the open device handle.

The contents is returned in a unsigned short buffer pointed by lpOutBuffer. The buffer must be always an unsigned short type independent of the TDRV003 variant. The argument nOutBufferSize specifies the size (size of USHORT) of the read buffer.

For the TPMC670 variants -11/-21 only the lower 8 bits are relevant.

Example

```
#include “tdrv003.h”

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

success = DeviceIoControl (
     hDevice,               //  TDRV003 handle
     IOCTL_TDRV003_READ,    //  control code
     NULL,
     0,
     &PortData,
     sizeof(PortData),
     &NumBytes,
     NULL                   // not over lapped
);

if( success ) {
     printf ("Read input port successful (input port = 0x%x)\n",
          PortData);
}
else {
     ErrorHandler ("Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_PARAMETER	This error is returned if the size of the read buffer is too small.

All other returned error codes are system error conditions.

4.1.3.3 IOCTL_TDRV003_READ_EVENT

The "event read function" attempts to read the contents of the input port either immediately or after a specified event occur.

Possible events are rising or falling edge or both, at a specified input bit or a pattern match of masked input bits.

Both parameter lpInBuffer and lpOutBuffer must pass a pointer to the read buffer *(TDRV003_READ_BUFFER*) to the device driver.

Data structure *TDRV003_READ_BUFFER*:

```
typedef struct {
        unsigned short      mode;
        unsigned short      mask;
        unsigned short      match;
        long                timeout;
} TDRV003_READ_BUFFER, *PTDRV003_ READ_BUFFER;
```

Members

mode

Specifies the “event” mode for this read request.

TDRV003_MATCH
The driver reads the input port if the masked input bits match to the specified pattern. The input mask must be specified in the parameter mask. A 1 value in mask means than the input bit value “must-match” identically to the corresponding bit in the match parameter.

TDRV003_HIGH_TR
The driver reads the input port if a high-transition at the specified bit position occur. A 1 value in mask specifies the bit position of the input port. If you specify more than one bit position the events are OR’ed. That means the read is completed if a high-transition at least at one relevant bit position occur.

TDRV003_LOW_TR
The driver reads the input port if a low-transition at the specified bit position occur. A 1 value in mask specifies the bit position of the input port. If you specify more than one bit position the events are OR’ed. That means the read is completed if a low-transition at least at one relevant bit position occur.

TDRV003_ANY_TR
The driver reads the input port if a transition (high or low) at the specified bit position occur. A 1 value in mask specifies the bit position of the input port. If you specify more than one bit position the events are OR’ed. That means the read is completed if a transition at least at one relevant bit position occur.

mask

Specifies a bit mask. A 1 value marks the corresponding bit position as relevant.

match

Specifies a pattern that must match to the contents of the input port. Only the bit positions specified by mask must compare to the input port.

timeout

Specifies the amount of time (in seconds) the caller is willing to wait for the specified event to occur. A value of 0 means wait indefinitely.

***TDRV003_MATCH* may miss very short events, because there is a latency between occurrence of the event and the ISR execution which checks the event. Therefore it is recommended not to use this event for fast changing inputs.**

Example

```
#include “tdrv003.h”

HANDLE    hDevice;
BOOLEAN success;
ULONG NumBytes;
TDRV003_READ_BUFFER ReadBuf;

/*
**   Read the input port after..
**   bit 0 = 0
**   bit 1 = 1
**   bit 6 = 0
**   bit 7 = 1
*/
ReadBuf.mode       = TDRV003_MATCH;
ReadBuf.mask       = 0x00C3;      // bit 0,1,6,7 are relevant
ReadBuf.match      = 0x0082;
ReadBuf.timeout    = 10;          // seconds
success = DeviceIoControl (
     hDevice,                     // TDRV003 handle
     IOCTL_TDRV003_READ_EVENT,
     &ReadBuf,                    // parameter for the driver
     sizeof(TDRV003_READ_BUFFER),
     &ReadBuf,                    // contains the read data
     sizeof(TDRV003_READ_BUFFER),
     &NumBytes,                   // size of returned ReadBuffer
     0
);

…
```

```
if( success ) {
     printf(“Success\n”);
}
else {
     ErrorHandler ("Device I/O control error”); // process error
}

/*
**   Read the input port after a high-transition at bit 7
**   occured
*/
ReadBuf.mode       = TDRV003_HIGH_TR;
ReadBuf.mask       = 1<<7;       // high-transition at bit 7
ReadBuf.timeout    = 10;         // seconds
success = DeviceIoControl (
     hDevice,                    // TDRV003 handle
     IOCTL_TDRV003_READ_EVENT,
     &ReadBuf,                   // parameter for the driver
     sizeof(TDRV003_READ_BUFFER),
     &ReadBuf,                   // contains the read data
     sizeof(TDRV003_READ_BUFFER),
     &NumBytes,                  // size of returned ReadBuffer
     0
);
if( success ) {
     printf(“Success\n”);
}
else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

ERROR_INVALID_PARAMETER	This error is returned if the size of the read buffer is too small or if the parameter mode contains an invalid value.
ERROR_NO_SYSTEM_RESOURCES	No more free entries in the drivers queue to handle concurrent event-controlled read requests. Increase the queue size (see also 3.1).
ERROR_SEM_TIMEOUT	The requested event does not occur within the specified time (timeout).

All other returned error codes are system error conditions.

4.1.3.4 IOCTL_TDRV003_WDENABLE

This control function enables the output watchdog function of the TDRV003 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_TDRV003_WDRESET* must be executed.

No additional parameter is required for this function.

Example

```
#include “tdrv003.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,                    //  TDRV003 handle
     IOCTL_TDRV003_WDENABLE,     //  control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                        //  not over lapped
);

if( success ) {
     printf("Enable output watchdog successful\n");
}
else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

No driver specific error codes

4.1.3.5 IOCTL_TDRV003_WDDISABLE

This device control function disables the output watchdog function of the TDRV003 enabled by *IOCTL_TDRV003_WDENABLE*.

No additional parameter is required for this function.

Example

```
#include “tdrv003.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,                    //  TDRV003 handle
     IOCTL_TDRV003_WDDISABLE,    //  control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                        //  not over lapped
);

if( success ) {
     printf("Disable output watchdog successful\n");
}
else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

No driver specific error codes

4.1.3.6 IOCTL_TDRV003_WDRESET

This device control function resets an output watchdog error. If *IOCTL_TDRV003_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 function.

Example

```
#include “tdrv003.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,                    //  TDRV003 handle
     IOCTL_TDRV003_WDRESET,      //  control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                        //  not over lapped
);

if( success ) {
     printf("Reset output watchdog successful\n");
}else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

No driver specific error codes

4.1.3.7 IOCTL_TDRV003_DEBENABLE

This control function enables the input debounce function.

The new timer value is passed by an unsigned short variable, pointed by lpInBuffer, to the driver. The argument nInBufferSize specifies the size (size of USHORT) of the.

See also TDRV003 Hardware User Manual – Debounce Time Register for counter calculation formulas.

Example

```
#include “tdrv003.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
USHORT    DebounceTime;


/*
**  Enable the debouncer with a debounce time of 1ms
*/
DebounceTime = 147;

success = DeviceIoControl (
     hDevice,                    //  TDRV003 handle
     IOCTL_TDRV003_DEBENABLE,    //  control code
     &DebounceTime,
     sizeof(DebounceTime),
     NULL,
     0,
     &NumBytes,
     NULL                        //  not overlapped
);


if( success ) {
     printf("Enable output watchdog successful\n");
}else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

ERROR_INVALID_PARAMETER	This error is returned if the size of the timer value buffer is too small

All other returned error codes are system error conditions.

4.1.3.8 IOCTL_TDRV003_DEBDISABLE

This control function disables the input debouncer function enabled by *IOCTL_TDRV003_DEBENABLE*.

No additional parameter is required for this function.

Example

```
#include “tdrv003.h”

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;


success = DeviceIoControl (
     hDevice,                        //  TDRV003 handle
     IOCTL_TDRV003_DEBDISABLE,       //  control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                            //  not over lapped
);

if( success ) {
     printf("Disable output watchdog successful\n");
}
else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

No driver specific error codes