The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV008-SW-65

Windows 2000/XP Device Driver

3x 16 bit I/O Ports with 512 Word FIFO and Handshake

Version 1.0.x

User Manual

Version 1.0.0

March 2007

powerBridge Computer

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

www.powerbridge.de
info@powerbridge.de

-(0)4101-4058-0
)4101-4058-19
@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

TDRV008-SW-65

Windows 2000/XP Device Driver

3 x 16 bit I/O Ports with 512 Word FIFO and Handshake

Supported Hardware Modules:

TPMC682

Issue	Description	Date
1.0.0	First Issue	March 21, 2007

Table of Content

1 Introduction

The TDRV008-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TDRV008 supported devices on an Intel or Intel-compatible x86 Windows 2000, Windows XP operating systems.

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 TDRV008 device driver supports the following features:

- write to the 8-bit GPO port 5
- read from the 8-bit GPI port 4
- configure ports (direction, mode and hardware timeout)
- buffered read and write of the 16-bit ports (0, 1 & 2) in pulsed or interlocked handshake mode
- hardware FIFO flush

Supported Modules:

TPMC682	3 x 16 bit I/O Ports with 512 Word FIFO and Handshake	PMC

In this document all supported modules and devices will be called TDRV008. Specials for certain devices will be advised.

To get more information about the features and use of TPMC682 devices it is recommended to read the manuals listed below.

TPMC682 User manual

TPMC682 Engineering Manual

2 Installation

Following files are located on the distribution media:

tdrv008.sys	Windows NT driver binary
tdrv008.h	Header-file with IOCTL code definitions
tdrv008.inf	Windows NT installation script
TDRV008-SW-65-1.0.0.pdf	This document
\example\tdrv008exa.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 TDRV008 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV008 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 TDRV008 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 (tdrv008.h, TDRV008-SW-65.pdf) to the desired target directories.

After successful installation the TDRV008 device driver will start immediately and creates devices (TDRV008_1, TDRV008_2 ...) for all recognized TDRV008 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 TDRV008 Buffered Digital I/O with Handshake (TPMC682)***" should appear.

3 TDRV008 Device Driver Programming

The TDRV008-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.)

3.1 TDRV008 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TDRV008 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TDRV008 Device

Before you can perform any I/O the *TDRV008* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TDRV008* 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 TDRV008 device to open. The *lpFileName* string should be of the form **\\.\TDRV008_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TDRV008_1, the second \\.\TDRV008_2 and so on.

DWORD dwDesiredAccess

This parameter specifies the type of access to the TDRV008.
For the TDRV008 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 TDRV008 devices.

DWORD dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TDRV008 devices must be always opened **OPEN_EXISTING**.

DWORD dwFlagsAndAttributes

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

HANDLE hTemplateFile

This value must be NULL for TDRV008 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TDRV008 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TDRV008_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,               // no security attrs
     OPEN_EXISTING,      // TDRV008 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 TDRV008 Device

The **CloseHandle** function closes an open TDRV008 handle.

BOOL CloseHandle(
HANDLE *hDevice;*
);

Parameters

HANDLE hDevice

Identifies an open TDRV008 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 ()

3.1.3 TDRV008 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

HANDLE hDevice

Handle to the TDRV008 that is to perform the operation.

DWORD 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_TDRV008_READ	Buffered read from a 16-bit handshake port
IOCTL_TDRV008_WRITE	Buffered write to a 16-bit handshake port
IOCTL_TDRV008_GETPORT	Get the state of the 8-bit GPI port #4
IOCTL_TDRV008_SETPORT	Set the state of the 8-bit GPO port #5
IOCTL_TDRV008_CONFPORT	Port setup
IOCTL_TDRV008_FLUSHPORTS	Flush the hardware FIFOs of the 16-bit ports

See behind for more detailed information on each control code.

LPVOID lpInBuffer

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

DWORD nInBufferSize

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

LPVOID lpOutBuffer

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

DWORD nOutBufferSize

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

LPDWORD 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 lpOverlapped

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

To use these TDRV008 specific control codes the header file tdrv008.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 TDRV008 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 ()

3.1.3.1 IOCTL_TDRV008_READ

This TDRV008 control function reads 16-bit values from the FIFO of a given port. If data isn't available when calling this function a timeout is used to implement a blocking read. A pointer to the I/O buffer (*TDRV008_RW_BUFFER*) is passed by the parameter *lpInBuffer* and additionally by *lpOutBuffer* to the driver.

The *TDRV008_RW_BUFFER* structure has the following layout:

```
typedef struct
{
    int             portNo;
    ULONG           flags;
    ULONG           timeout;
    int             bufferSize;
    int             validWords;
    USHORT          data[TDRV008_FIFOSIZE];

} TDRV008_RW_BUFFER, *PTDRV008_RW_BUFFER;
```

Members

portNo

This parameter holds the port number to read from. Valid values are 0, 1 and 2.

flags

This parameter decides about non-blocking and blocking read requests. For non-blocking operation set *flags* to TDRV008_F_RW_NOWAIT. To initialize a blocking read request set *flags* to zero.

timeout

This parameter defines the read timeout with a given resolution of one second. So if you set it to 1 the driver should wait at least one second and worst case two seconds before terminating the certain read request.

bufferSize

This parameter defines the maximum count of words to read. The *data* storage has to be large enough to receive the specified amount of 16-bit words.

validWords

This in and out parameter holds the count of data words actually read. After read requests completion maybe not all data words given by *bufferSize* were received in the given time. For this purpose *validWords* is used for the read request result. If it is set to a value greater than 0 when starting the read request and blocking read is used then *validWords* is meant as an offset to the first element of the *data* buffer. So successive read requests that are partially completed can use the same I/O buffer until final read request completion.

data

This field parameter is used as data storage. It has a fixed size of TDRV008_FIFOSIZE words to match the hardware fifo size.

Example

```
#include "tdrv008.h"

...
HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TDRV008_RW_BUFFER rwBuf;
int i;

rwBuf.portNo = 1;                  // read from port 1
rwBuf.flags = 0;                   // blocking read
rwBuf.timeout = 5;                 // wait at least 5 seconds
rwBuf.bufferSize = 177;            // we want to receive 177 words
rwBuf.validWords = 0;              // no data received yet
/* rwBuf.data[] will be filled with incoming data words */

success = DeviceIoControl (
     hDevice,                 // TDRV008 device handle
     IOCTL_TDRV008_READ,      // control code
     &rwBuf,                  // input buffer
     sizeof(rwBuf),
     &rwBuf,                  // and output buffer point to rwBuf
     sizeof(rwBuf),
     &NumBytes,               // number of bytes transferred
     NULL
);

if( success ) {
     // Process data, rwBuf.validWords is the real read result
     for (i = 0; i < rwBuf.validWords; i++)
     {
          printf("@0x&04X: 0x%02X", i, rwBuf.data[i]);
     }
     ...
}
else {
     // Process DeviceIoControl() error
     ...
}
...
```

Error Codes

ERROR_INVALID_PARAMETER	The size of the message buffer is too small or invalid I/O buffer member (bufferSize, validWords, ...) .
ERROR_NO_SUCH_DEVICE	The port specified by I/O buffer member portNo doesn't exist.
ERROR_INVALID_DEVICE_STATE	The port was configured as output port. It has to be an input port to start a read request.
ERROR_NETWORK_BUSY	The certain port is in use.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.2 IOCTL_TDRV008_WRITE

This TDRV008 control function writes 16-bit values to the specified buffered output port. A pointer to an I/O buffer structure (*TDRV008_RW_BUFFER*) is passed by the parameter *lpInBuffer* and *lpOutBuffer* to the driver.

The *TDRV008_RW_BUFFER* structure has the following layout:

```
typedef struct
{
    int             portNo;
    ULONG           flags;
    ULONG           timeout;
    int             bufferSize;
    int             validWords;
    USHORT          data[TDRV008_FIFOSIZE];

} TDRV008_RW_BUFFER, *PTDRV008_RW_BUFFER;
```

Members

portNo

This parameter holds the port number of the handshake port to write on. Valid values are 0, 1 and 2.

flags

This parameter is not used for this IOCTL function.

timeout

This parameter defines the write timeout with a given resolution of one second. So if you set it to 1 the driver should wait at least one second and worst case two seconds before terminating the certain write request.

bufferSize

This parameter defines the count of words to write to the hardware FIFO of the certain handshake port.

validWords

This in and out parameter holds the count of data words actually written. In the case of a full FIFO the write process can't send more data words to the certain port and will block for *timeout* seconds. For this purpose *validWords* is used for the write request result. If it is set to a value greater than 0 when starting the write request then *validWords* is meant as an offset to the first element of the *data* buffer. So successive write requests that are partially completed can use the same I/O buffer until final write request completion.

data

This field parameter is used as data source during the handshake transmission. It has a fixed maximum size of TDRV008_FIFOSIZE words to match the hardware fifo size.

Example

```
#include "tdrv008.h"

...

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TDRV008_RW_BUFFER rwBuf;
int i;

rwBuf.portNo = 0;                // read from port 1
rwBuf.timeout = 2;               // wait at least 2 seconds
rwBuf.bufferSize = 411;          // we want to send 411 words
rwBuf.validWords = 0;            // start with element 0

for (i = 0; i < rwBuf.bufferSize; i++)
{
     rwBuf.data[i] = ...;        // user data
}

success = DeviceIoControl (
     hDevice,                // TDRV008 device handle
     IOCTL_TDRV008_WRITE,    // control code
     &rwBuf,                 // input buffer
     sizeof(rwBuf),
     &rwBuf,                 // and output buffer point to rwBuf
     sizeof(rwBuf),
     &NumBytes,              // number of bytes transferred (ignore)
     NULL
);

...
```

```
...

if( success ) {
     // Process data, rwBuf.validWords is the real write result
     for (i = 0; i < rwBuf.validWords; i++)
     {
          printf("@0x&04X: 0x%02X", i, rwBuf.data[i]);
     }
     ...
}
else {
     // Process DeviceIoControl() error
     ...
}
...
```

Error Codes

ERROR_INVALID_PARAMETER	The size of the message buffer is too small or invalid I/O buffer member (bufferSize, validWords, ...) .
ERROR_NO_SUCH_DEVICE	The port specified by I/O buffer member portNo doesn't exist.
ERROR_INVALID_DEVICE_STATE	The port was configured as input port. It has to be an output port to start a write request.
ERROR_NETWORK_BUSY	The certain port is in use.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.3 IOCTL_TDRV008_GETPORT

This TDRV008 control function reads the state of the free input lines of the 8 bit general purpose port 4. Only the upper 5 bit of the value are valid the lower 3 bits will always be set to 0. A pointer to the input buffer (*UCHAR*) is passed by the parameter *lpInBuffer* to the driver. The *lpOutBuffer* is not used and should be a *NULL* pointer.

Example

```
#include "tdrv008.h"

...

HANDLE hDevice;
BOOLEAN success;
ULONG NumRead;
UCHAR ucVal;

success = DeviceIoControl(  hCurrent,               //  TDRV008 handle
                            IOCTL_TDRV008_GETPORT,  //  control code
                            NULL,
                            0,
                            &ucVal,
                            sizeof(ucVal),
                            &NumRead,
                            NULL);
//
//  Check the result of the last device I/O control operation
//
if( success )
{
     printf("OK\n");
     printf("    port4 (bit7..3): %02Xh\n", ucVal);
     ...
}
else
{
     printf( "\nReading port4 failed -->  Error = %d\n", GetLastError() );
      ...
}
...
```

Error Codes

ERROR_INVALID_PARAMETER	The size of the message buffer is too small.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.4 IOCTL_TDRV008_SETPORT

This TDRV008 control function sets the state of the free output lines of the general purpose port 5. Only the upper 5 bit of the value are valid the lower 3 bits are ignored. A pointer to the output buffer (*UCHAR*) is passed by the parameter *lpOutBuffer* to the driver. The *lpInBuffer* is not used and should be a *NULL* pointer.

Example

```
#include "tdrv008.h"

...

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
UCHAR ucVal;

ucVal = 0x42;      // bits 0..2 are ignored -> so 0x40 will be written

success = DeviceIoControl(  hCurrent,               //  TDRV008 handle
                            IOCTL_TDRV008_SETPORT,  //  control code
                            &ucVal,
                            sizeof(ucVal),
                            NULL,
                            0,
                            &NumWritten,
                            NULL);
//
//  Check the result of the last device I/O control operation
//
if( success )
{
     printf("OK\n");
     ...
}
else
{
     printf( "\nWriting port5 failed -->  Error = %d\n", GetLastError() );
     ...
}
...
```

Error Codes

ERROR_INVALID_PARAMETER	The size of the message buffer is too small.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.5 IOCTL_TDRV008_CONFPORT

This TDRV008 control function configures a specified handshake port. The pointer to the configuration buffer (*TDRV008_CONF_BUFFER*) is passed by the parameter *lpOutBuffer* to the driver. The parameter *lpInBuffer* is not used and should be set to NULL.

The *TDRV008_CONF_BUFFER* structure has the following layout:

```
typedef struct
{

    int             portNo;
    ULONG           flags;
    BOOLEAN         enaOutput;
    USHORT          fifoTimeout;
    USHORT          fifoThreshold;

} TDRV008_CONF_BUFFER, *PTDRV008_CONF_BUFFER;
```

Members

portNo

This parameter specifies the handshake port. Valid values are 0, 1 and 2.

flags

This parameter specifies the output handshake mode. (Refer to the User Manual of your module for a detailed description of the output handshake modes). Following values are valid.

TDRV008_F_CONF_HOUT_HSNONE	No output handshake
TDRV008_F_CONF_HOUT_HSINTERLOCKED	Interlocked output handshake
TDRV008_F_CONF_HOUT_HSPULSED	Pulsed output handshake

enaOutput

This parameter defines the direction of the port. If this parameter is set *TRUE* the port will be configured as an output port, if it is specified *FALSE* the port will be configured as input.

fifoTimeout

This parameter specifies the hardware FIFO timeout value. The value will be directly written to the module (Register TCPRx - refer to the User Manual of your module for more information). This value is only used for input ports.

fifoThreshold

This parameter specifies the FIFO threshold value. This value will be directly written to the module (Register FIFO_FTRx - refer to the User Manual of your module for more information). Valid values 1 to 512.

EXAMPLE

```
#include "tdrv008.h"

...

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
UCHAR ucVal;
TDRV008_CONF_BUFFER confBuf;

...

/* Setup handshake port 0 */
/* - output */
/* - interlocked output handshake */
/* - threshold: 256 */
confBuf.portNo =   0;
confBuf.flags =    TDRV008_F_CONF_HOUT_HSINTERLOCKED;
confBuf.enaOutput =     TRUE;
confBuf.fifoTimeout =   0;   /* not used */
confBuf.fifoThreshold = 256;

printf("\nConfigure port ... ");
success = DeviceIoControl(  hCurrent,                //  TDRV008 handle
                            IOCTL_TDRV008_CONFPORT,  //  control code
                            &confBuf,
                            sizeof(confBuf),
                            NULL,
                            0,
                            &NumWritten,
                            NULL);

...
```

```
...

//
//  Check the result of the last device I/O control operation
//
if( success )
{
     printf("OK\n");
}
else
{
     printf( "\nIOCTL failed -->  Error = %d\n", GetLastError() );
}
...
```

Error Codes

ERROR_INVALID_PARAMETER	The size of the config buffer is too small or invalid config buffer member (flags, ...) .
ERROR_NO_SUCH_DEVICE	The port specified by config buffer member portNo doesn't exist.
ERROR_NETWORK_BUSY	The certain port is in use.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.6 IOCTL_TDRV008_FLUSHPORTS

This TDRV008 control function flushes the FIFOs of all handshake ports (0, 1, and 2). This may be useful after configuration. The parameter pointer lpOutBuffer and lpInBuffer are not used and should be set to NULL.

EXAMPLE

```
#include "tdrv008.h"

...
HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;

printf("\nFlush ports ... ");
success = DeviceIoControl(  hCurrent,                   //  TDRV008 handle
                            IOCTL_TDRV008_FLUSHPORTS,   //  control code
                            NULL,
                            0,
                            NULL,
                            0,
                            &ioctlReturn,
                            NULL);
//
//  Check the result of the last device I/O control operation
//
if( success )
{
     printf("OK\n");
}
else
{
     printf( "\nIOCTL failed -->  Error = %d\n", GetLastError() );
}
```

Error Codes

All returned error codes are system error conditions. There are control function specific error codes.

See Also

Win32 documentation DeviceIoControl()