The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV006-SW-65

Windows 2000/XP Device Driver

64 Channel Digital I/O

Version 1.0.x

User Manual

Issue 1.0.1

September 2006

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

TDRV006-SW-65

Windows 2000/XP Device Driver

64 Channel Digital I/O

Supported Modules:
TPMC681

Issue	Description	Date
1.0.0	First Issue	March 31, 2006
1.0.1	New Address TEWS LLC	September 20, 2006

Table of Content

1 Introduction

The TDRV006-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 TDRV006-SW-65 device driver supports the following features:

- read actual input value
- write new output value
- configure I/O lines for input or output
- wait for input events

The TDRV006-SW-65 device driver supports the modules listed below:

TPMC681	64 I/O Lines	PMC

To get more information about the features and use of TDRV006 devices it is recommended to read the manuals listed below.

TPMC681 User manual

TPMC681 Engineering Manual

2 Installation

Following files are located on the distribution media:

tdrv006.sys	TDRV006 device driver
tdrv006.h	TDRV006 include file for application
tdrv006.inf	TDRV006 windows installation script
example\tdrv006exa.c	Example application
TDRV006-SW-65-1.0.0.pdf	PDF copy of this manual
Release.txt	Release information

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV006 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV006 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 TDRV006 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 (tdrv006.h, …) to the desired target directories.

After successful installation the TDRV006 device driver will start immediately and creates devices (TDRV006_1, TDRV006_2 ...) for all recognized TDRV006 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 TDRV006 (Digital I/O)***" should appear.

3 TDRV006 Device Driver Programming

The TDRV006-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 TDRV006 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TDRV006 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TDRV006 Device

Before you can perform any I/O the *TDRV006* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TDRV006* 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 TDRV006 to open. The *lpFileName* string should be of the form **\\.\TDRV006_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TDRV006_1, the second \\.\TDRV006_2 and so on.

DWORD dwDesiredAccess

This parameter specifies the type of access to the TDRV006.
For the TDRV006 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 TDRV006 devices.

DWORD dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TDRV006 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 TDRV006 devices.

HANDLE hTemplateFile

This value must be NULL for TDRV006 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TDRV006 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TDRV006_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                    // no security attrs
     OPEN_EXISTING,           // TDRV006 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()

3.1.2 Closing a TDRV006 Device

The **CloseHandle** function closes an open TDRV006 handle.

```
BOOL CloseHandle(
      HANDLE hDevice;
);
```

Parameters

HANDLE hDevice

Identifies an open TDRV006 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 TDRV006 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 TDRV006 device 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 *tdrv006.h*:

Value	Meaning
IOCTL_TDRV006_WRITE	Write output port
IOCTL_TDRV006_READ	Read input port immediately
IOCTL_TDRV006_CONFIG	Configure I/O line directions
IOCTL_TDRV006_WAIT_EVENT	Read input port after specified event occur

See below for more detailed information on each control code.

To use these TDRV006 specific control codes the header file *tdrv006.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()

3.1.3.1 IOCTL_TDRV006_WRITE

This control function writes the specified value to the output port of the TDRV006 device associated with the open device handle.

The new port value is passed in a buffer (*TDRV006_PARAM_BUFFER*) pointed to by *lpInBuffer,* to the driver. The argument *nInBufferSize* specifies the size (*sizeof(TDRV006_PARAM_BUFFER)*) of the buffer.

The *TDRV006_PARAM_BUFFER* structure has the following layout:

```
typedef struct
{
        unsigned long       value_0_31;
        unsigned long       value_32_63;
} TDRV006_PARAM_BUFFER, *PTDRV006_PARAM_BUFFER;
```

value_0_31

This value specifies the new output value for I/O lines 0 up to 31. Bit 0 of the value specifies the value for I/O line 0, bit 1 for I/O line 1 and so on.

value_32_63

This value specifies the new output value for I/O lines 32 up to 63. Bit 0 of the value specifies the value for I/O line 32, bit 1 for I/O line 33 and so on.

The specified output value is written to the output registers of the device, but only the I/O lines configured for output will be set. If the I/O line direction is set to output the last written value of the output register is used.

Example

```
#include <windows.h>
#include <winioctl.h>
#include "tdrv006.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TDRV006_PARAM_BUFFER    PortData;

/* set I/O lines 0..3, 8, 48..63 to HIGH */
/* all other I/O lines shall be set to LOW */
PortData.value_0_31 =   0x0000010F;
PortData.value_32_64 =  0xFFFF0000;

…
```

…

```
success = DeviceIoControl (
     hDevice,               // TDRV006 device handle
     IOCTL_TDRV006_WRITE,   // control code
     &PortData,
     sizeof(PortData),
     NULL,
     0,
     &NumBytes,
     NULL                   // not overlapped
);

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

Error Codes

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

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.2 IOCTL_TDRV006_READ

This control function reads the value of the input register of the TDRV006 device associated with the open device handle.

The port value is returned in the read buffer (*TDRV006_PARAM_BUFFER*) pointed to by *lpOutBuffer*. The argument *nOutBufferSize* specifies the size (*sizeof(TDRV006_PARAM_BUFFER)*) of the read buffer.

The *TDRV006_PARAM_BUFFER* structure has the following layout:

```
typedef struct
{
        unsigned long       value_0_31;
        unsigned long       value_32_63;
} TDRV006_PARAM_BUFFER, *PTDRV006_PARAM_BUFFER;
```

value_0_31

This value will be filled with the value of input register for I/O lines 0 up to 31. Bit 0 of the value returns the value for I/O line 0, bit 1 for I/O line 1 and so on.

value_32_63

This value will be filled with the value of input register for I/O lines 32 up to 63. Bit 0 of the value returns the value for I/O line 32, bit 1 for I/O line 33 and so on.

Example

```
#include <windows.h>
#include <winioctl.h>
#include "tdrv006.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TDRV006_PARAM_BUFFER    PortData;

success = DeviceIoControl (
     hDevice,               // TDRV006 device handle
     IOCTL_TDRV006_READ,    // control code
     NULL,
     0,
     &PortData,
     sizeof(PortData),
     &NumBytes,
     NULL                   // not overlapped
);

…
```

…

```
if( success ) {
     printf("\nRead input value successful\n");
     printf("    Input value: %lXh - %lXh\n",
               PortData.value_32_63", PortData.value_0_31);
}
else {
     ErrorHandler("Device I/O control 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.

See Also

Win32 documentation DeviceIoControl()

3.1.3.3 IOCTL_TDRV006_CONFIG

This control function configures the direction of the I/O lines of the TDRV006 device associated with the open device handle.

The direction is passed in a buffer (*TDRV006_PARAM_BUFFER*) pointed to by *lpInBuffer* to the driver. The argument *nInBufferSize* specifies the size (*sizeof(TDRV006_PARAM_BUFFER)*) of the buffer.

The *TDRV006_PARAM_BUFFER* structure has the following layout:

```
typedef struct
{
        unsigned long       value_0_31;
        unsigned long       value_32_63;
} TDRV006_PARAM_BUFFER, *PTDRV006_PARAM_BUFFER;
```

value_0_31

This value specifies the direction for I/O lines 0 up to 31. Bit 0 of the value specifies the direction for I/O line 0, bit 1 for I/O line 1 and so on.

bit value	direction
0	configure I/O line for input
1	configure I/O line for output

value_32_63

This value specifies the direction for I/O lines 32 up to 33. Bit 0 of the value specifies the direction for I/O line 32, bit 1 for I/O line 63and so on.

bit value	direction
0	configure I/O line for input
1	configure I/O line for output

After setting the configuration of an I/O line to output, the pin will be set to the level actually specified in the output register.

Example

```
#include <windows.h>
#include <winioctl.h>
#include "tdrv006.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TDRV006_PARAM_BUFFER    PortData;

…
```

…

```
/* configure I/O lines 0..24 for output, the other lines shall be input */
PortData.value_0_31 =   0x00FFFFFF;
PortData.value_32_64 =  0x00000000;

success = DeviceIoControl (
     hDevice,                // TDRV006 device handle
     IOCTL_TDRV006_CONFIG,   // control code
     &PortData,
     sizeof(PortData),
     NULL,
     0,
     &NumBytes,
     NULL                    // not overlapped
);

if( success ) {
     printf("\nDirection setting successful\n");
}
else {
     ErrorHandler("Device I/O control error”);
}
```

Error Codes

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

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

3.1.3.4 IOCTL_TDRV006_WAIT_EVENT

This function waits for a specified event of a specified I/O line of the TDRV006 device associated with the open device handle

The event parameters are passed in a buffer (*TDRV006_WAIT_BUFFER*) pointed to by *lpInBuffer* to the driver. The argument *nInBufferSize* specifies the size (*sizeof(TDRV006_WAIT_BUFFER)*) of the buffer.

The *TDRV006_WAIT_BUFFER* structure has the following layout:

```
typedef struct
{
        unsigned long       mode;
        unsigned long       inpLine;
        long                timeout;
} TDRV006_WAIT_BUFFER, *PTDRV006_WAIT_BUFFER;
```

mode

This parameter specifies the event to wait for. The following events can be selected with the predefined values (refer to *tdrv006.h*):

event value	description
TDRV006_HIGH_TR	The function returns if a low to high transition (positive edge) occurs on the specified I/O line.
TDRV006_LOW_TR	The function returns if a high to low transition (negative edge) occurs on the specified I/O line.
TDRV006_ANY_TR	The function returns if a transition (positive or negative edge) occurs on the specified I/O line.

inpLine

This parameter specifies the I/O line on which the event shall occur. Valid values are 0 up to 63.

timeout

This value specifies the maximum time to wait for the event. If this specified time has passed the function will return indicating an error. The time is specified in steps of seconds. Specifying a 0 means the function will never timeout (wait forever).

There is an incorrectness of about 1 second in the specified timeout. Specifying a timeout value of 10 seconds means that the function may return after 9 up to 10 seconds.

Example

```
#include <windows.h>
#include <winioctl.h>
#include "tdrv006.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TDRV006_WAIT_BUFFER     WaitBuf;

/* Wait for a Low to High transition on I/O line 17 */
WaitBuf.mode       = TDRV006_HIGH_TR;
WaitBuf.inpLine    = 17;
WaitBuf.timeout    = 10;          // 9..10 seconds
success = DeviceIoControl (
     hDevice,                     // TDRV006 device handle
     IOCTL_TDRV006_WAIT_EVENT,    // control code
     &WaitBuf,
     sizeof(WaitBuf),
     NULL,
     0,
     &NumBytes,
     0                            // not overlapped
);
if( success ) {
     printf("Event occurred\n");
}
else {
     ErrorHandler ("Device I/O control error”);
}
```

Error Codes

ERROR_INVALID_PARAMETER	This error is returned if the size of the wait buffer is too small. The specified I/O line or mode is invalid.
ERROR_NO_SYSTEM_RESOURCES	The specified I/O line is already in use to wait for an event.
ERROR_SEM_TIMEOUT	The specified event has not occurred within the specified timeout time.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()