The Embedded I/O Company

TEWS TECHNOLOGIES

TIP500-SW-65

Windows 2000/XP Device Driver

16 Channel 12-Bit ADC

Version 1.0.x

User Manual

Issue 1.0.0

December 2005

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

TEWS TECHNOLOGIES LLC
1 E. Liberty Street, Sixth Floor
Reno, Nevada 89504 / USA
Phone: +1 (775) 686 6077
Fax: +1 (775) 686 6024
e-mail: usasales@tews.com
www.tews.com

TIP500-SW-65

16 Channel 12-Bit ADC

Windows 2000/XP Device Driver

Issue	Description	Date
1.0.0	First Issue	December 1, 2005

Table of Contents

1 Introduction

The TIP500-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TIP500 on Intel or Intel-compatible x86 Windows 2000 or 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 device handle and for performing device I/O control operations.

Because the TIP500 device driver is stacked on the TEWS TECHNOLOGIES IPAC carrier driver, it's necessary to install also the appropriate IPAC carrier driver. Please refer to the IPAC carrier driver user manual for further information.

The TIP500 device driver includes the following functions:

- reading analog input values from specified channels
- standard and pipeline mode
- single-ended and differential input mode
- automatic offset and gain correction with factory calibration data

To understand all features of this device driver, it is recommended to read the TIP500 User Manual.

2 Installation

Following files are located on the distribution disk:

tip500.sys	Device driver binary
tip500.h	Header file with IOCTL code definitions and driver specific data types
tip500.inf	Installation script
TIP500-SW-65-1.0.0.pdf	This document in PDF format
\example\tip500exa.c	example application C source file
Release.txt	Release information

2.1 Software Installation

The TIP500 Device Driver software assumes a correctly installed and active IPAC carrier driver.

2.1.1 Windows 2000/XP

This section describes how to install the TIP500 Device Driver on a Windows 2000/XP operating system.

After installing the TIP500 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 TIP500 driver disk; and select "***Disk Drive***" and/or "***CD-ROM***" 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. Completing the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Now copy all needed files (tip500.h, ...) to the desired target directories.

After successful installation the TIP500 device driver will start immediately and creates devices (TIP500_1, TIP500_2, ...) for all recognized TIP500 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 "***TIP500***" should appear.

3 TIP500 Device Driver Programming

The TIP500-SW-65 Windows 2000/XP 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 device 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 TIP500 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TIP500 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TIP500 Device

Before you can perform any I/O the *TIP500* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TIP500* device.

```
HANDLE CreateFile(
    LPCTSTR lpFileName,                          // pointer to filename
    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
);
```

Parameters

lpFileName

Points to a null-terminated string that specifies the name of the TIP500 to open.
The *lpFileName* string should be of the form **\\.\tip500_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\tip500_1, the second \\.\tip500_2 and so on.

dwDesiredAccess

Specifies the type of access to the TIP500. For the TIP500 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE).

dwShareMode

A set of bit flags that specifies how the object can be shared for read and write. Unimportant for TIP500, set to 0.

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TIP500 devices.

dwCreationDistribution

Specifies which action to take on files that exist and which action to take when files that do not exist. TIP500 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 0 for TIP500 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TIP500 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call **GetLastError**.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\tip500_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TIP500 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 TIP500 Device

The **CloseHandle** function closes an open TIP500 handle.

```
BOOL CloseHandle(
      HANDLE hDevice;                          // handle to a TIP500 device to close
);
```

Parameters

hDevice

Identifies an already opened TIP500 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 TIP500 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 TIP500 that is to perform the operation.

dwIoControlCode

Specifies the control code for an operation. This value identifies the specific operation to be performed. The following values are defined in *TIP500.h*:

Value	Meaning
IOCTL_TIP500_READ	Read analog input value
IOCTL_TIP500_INFO	Read device information data

See behind for more detailed information on each control code.

To use these TIP500 specific control codes the header file tip500.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***.

The 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_TIP500_READ

The read function starts an AD conversion at the specified input channel of the TIP500 and returns the converted analog value to the caller. A pointer to the read buffer structure (*T500_READ_BUFFER*) must be passed by the arguments *lpInBuffer* and *lpOutBuffer* to the driver. The arguments *nOutBufferSize* and *nInBufferSize* specifies the length of this buffer.

Before calling the read function some elements of the read buffer must be set to appropriate values (see below for a detailed description of each element). After successful execution the element *data* returns the converted analog input value as a two's complement integer value.

The read function will always use the fastest possible operating mode.

The *T500_READ_BUFFER* structure has the following layout:

```
typedef struct {
    int     chan;
    int     gain;
    int     flags;
    int     data;
} T500_READ_BUFFER;
```

chan

Specifies the channel number at which the conversion will be started. Valid channel numbers are 1...16 if single-ended is selected or 1...8 for differential input.

gain

Specifies the gain for the input voltage amplifier. Valid gain constants are listed below:

Definition	Gain	Valid for TIP500 Variant
T500_GAIN_1	1	TIP500-10/-11/-20/-21
T500_GAIN_2	2	TIP500-10/-11/-20/-21
T500_GAIN_4	4	TIP500-11/-21
T500_GAIN_5	5	TIP500-10/-20
T500_GAIN_8	8	TIP500-11/-21
T500_GAIN_10	10	TIP500-10/-20

flags

This bit mask controls the read operation; you create the value by the bitwise OR of the appropriate parameters (using the | operator in C).

T500_DIFF	Use differential analog inputs. If this flag is omitted single-ended inputs will be selected.
T500_PIPELINE	Use data pipeline mode. In this mode converted data from the previous conversion will be read and a new conversion is started. If this flag is omitted, data from the new started conversion will be read (see also the TIP500 User Manual for more information).
T500_CORRECTION	Perform an automatic offset and gain correction with factory calibration data stored in the TIP500 ID-PROM. If this flag is omitted the converted data will be read directly.

data

Converted analog input value (two's complement). The data read from the driver is a right aligned 12-bit signed value.

Range: -2048..2047 (for TIP500-10/-11)
Range: 0..4095 (for TIP500-20/-21)

Example

```
#include "tip500.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
T500_READ_BUFFER rdBuf;


rdBuf.chan    = 1;
rdBuf.gain    = T500_GAIN_1;
rdBuf.flags   = T500_DIFF | T500_CORRECTION;

success = DeviceIoControl (
     hDevice,               // TIP500 handle
     IOCTL_TIP500_READ,
     &rdBuf,                // parameter for the driver
     sizeof(T500_READ_BUFFER),
     &rdBuf,                // data from the driver
     sizeof(T500_READ_BUFFER),
     &NumBytes,             // size of returned Buffer
     0
);

if( !success )
{
     ErrorHandler ("Device I/O control error");  // process error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The input or output buffer is too small. Please check the parameters *nInBufferSize* and *nOutBufferSize* of the DeviceIoControl () function call.
ERROR_MEMBER_NOT_IN_GROUP	Invalid channel number.
ERROR_INVALID_PARAMETER	Some parameters are out of range or invalid (gain, mode or correction).
ERROR_SEM_TIMEOUT	Timeout occurred during conversion.

See Also

Win32 documentation DeviceIoControl(), TIP500 Hardware User Manual

3.1.3.2 IOCTL_TIP500_INFO

This control function reads the module variant and the factory calibration data from the specified device and returns this information in the *T500_INFO_BUFFER* structure to the caller.

A pointer to the *T500_INFO_BUFFER* structure is passed by the argument *lpOutBuffer* to the driver. The argument *nOutBufferSize* specifies the length of this buffer.

The *T500_INFO_BUFFER* structure has the following layout:

```
typedef struct
{
    int     variant;
    short   offset_corr[4];
    short   gain_corr[4];
} T500_INFO_BUFFER;
```

variant

Returns the module variant.

value	module variant
10	TIP500-10
11	TIP500-11
20	TIP500-20
21	TIP500-21

offset_corr

The factory programmed correction data for offset correction is returned in this array. The index of the array specifies the input gain. (See table below)

gain_corr

The factory programmed correction data for gain correction is returned in this array. The index of the array specifies the input gain. (See table below)

Index	Gain (TIP500-10/20)	Gain (TIP500-11/21)
0	1	1
1	2	2
2	5	4
3	10	8

Example

```
#include “tip500.h”


HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
T500_INFO_BUFFER  infoBuf;


success = DeviceIoControl (
     hDevice,                // TIP500 handle
     IOCTL_TIP500_INFO,
     NULL,                   // not used, set to NULL
     0,                      // not used, set to 0
     &infoBuf,
     sizeof(T500_INFO_BUFFER),
     &NumBytes,
     0
);


if( !success ) {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The output buffer is too small. Please check the argument *nOutBufferSize.*

See Also

Win32 documentation DeviceIoControl(), TIP500 Hardware User Manual