The Embedded I/O Company

TEWS TECHNOLOGIES

TIP550-SW-65

Windows 2000/XP Device Driver

8 (4) Channel 12 Bit D/A

Version 1.1.x

User Manual

Issue 1.1.0

June 2009

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

TIP550-SW-65

Windows 2000/XP Device Driver

8 (4) Channel 12 Bit D/A

Supported Modules:
TIP550

Issue	Description	Date
1.0	First Issue	March 3, 2004
1.1	WinXP description added, Win98/Me description removed	June 14, 2004
1.1.0	General Revision	June 9, 2009

Table of Contents

1 Introduction

1.1 Device Driver

The TIP550-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TIP550 on an Intel or Intel-compatible x86 Windows 2000/XP operating system.

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 TIP550 device driver is stacked on the TEWS TECHNOLOGIES IPAC Carrier Driver, it is necessary to install also the appropriate IPAC Carrier Driver. Please refer to the IPAC Carrier Driver user manual for further information.

The TIP550-SW-65 device driver includes the following functions:

- writing D/A output value
- data output correction using manufacturer stored data
- configure D/A channels

The TIP550-SW-65 device driver supports the modules listed below:

TIP550-10	8 Channel 12 bit-DAC	IndustryPack®
TIP550-11	4 Channel 12 bit-DAC	IndustryPack®

To get more information about the features and use of TIP550 devices it is recommended to read the manuals listed below.

TIP550 User manual

TIP550 Engineering Manual

1.2 IPAC Carrier Driver

IndustryPack (IPAC) carrier boards have different implementations of the system to IndustryPack bus bridge logic, different implementations of interrupt and error handling and so on. Also the different byte ordering (big-endian versus little-endian) of CPU boards will cause problems on accessing the IndustryPack I/O and memory spaces.

To simplify the implementation of IPAC device drivers which work with any supported carrier board, TEWS TECHNOLOGIES has designed a so called Carrier Driver that hides all differences of different carrier boards under a well defined interface.

The TEWS TECHNOLOGIES IPAC Carrier Driver CARRIER-SW-65 is part of this TIP550-SW-65 distribution. It is located in directory CARRIER-SW-65 on the corresponding distribution media.

This IPAC Device Driver requires a properly installed IPAC Carrier Driver. Due to the design of the Carrier Driver, it is sufficient to install the IPAC Carrier Driver once, even if multiple IPAC Device Drivers are used.

Please refer to the CARRIER-SW-65 User Manual for a detailed description how to install and setup the CARRIER-SW-65 device driver, and for a description of the TEWS TECHNOLOGIES IPAC Carrier Driver concept.

2 Installation

Following files are located on the distribution media:

Directory path 'TIP550-SW-65':

tip550.sys	Device driver binary
tip550.h	Header file with IOCTL code definitions and driver specific data types
tip550.inf	Installation script
TIP550-SW-65-1.1.0.pdf	This document in PDF format
example\tip550exa.c	example application C source file
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Software Installation

The TIP550-SW-65 Device Driver software assumes a correctly installed and active TEWS TECHNOLOGIES IPAC Carrier Driver.

2.1.1 Windows 2000/XP

This section describes how to install the TIP550 Device Driver on a Windows 2000/XP operating system.

After installing the TIP550 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 TIP550 driver media 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 media.
   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 (tip550.h, ...) to the desired target directories.

After successful installation the TIP550 device driver will start immediately and create devices (tip550_1, tip550_2, ...) for all recognized TIP550 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 - TIP550 (8/4 Channel - 12-Bit DAC)***" should appear.

3 TIP550 Device Driver Programming

The TIP550-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 TIP550 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TIP550 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TIP550 Device

Before you can perform any I/O the TIP550 device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the TIP550 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 TIP550 to open.
The *lpFileName* string should be of the form **\\.\tip550_*x*** to open the device *x.* The ending x is a one-based number. The first device found by the driver is \\.\tip550_1, the second \\.\tip550_2 and so on.

dwDesiredAccess

Specifies the type of access to the TIP550. For the TIP550 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. Not used for TIP550, set to 0.

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TIP550 devices.

dwCreationDistribution

Specifies which action to take on files that exist and which action to take when files that do not exist. TIP550 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 TIP550 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TIP550 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call **GetLastError**.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\tip550_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TIP550 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 TIP550 Device

The **CloseHandle** function closes an open TIP550 handle.

BOOL CloseHandle
(
HANDLE *hDevice;* // handle to a TIP550 device to close
)

Parameters

hDevice

Identifies an already opened TIP550 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 TIP550 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 TIP550 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 tip550.h:

Value	Meaning
IOCTL_TIP550_WRITE	Set output lines
IOCTL_TIP550_CONFIG	Configure voltage range selection

See behind for more detailed information on each control code.

To use these TIP550 specific control codes the header file tip550.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_TIP550_WRITE

This control function starts a D/A conversion on the specified channel.

The parameter *lpInBuffer* passes a pointer to an I/O buffer (*TIP550_IO_BUFFER*) to the driver which contains parameter required to perform the operation like channel number, automatic correction control and the DA value to output.

lpOutBuffer must pass a *NULL* pointer to the device driver.

Before calling this function the device must be configured by using the *IOCTL_TIP550_CONFIG* function.

```
typedef struct {
        ULONG           channel;
        ULONG           correction;
        LONG            data;
} TIP550_IO_BUFFER, *PTIP550_IO_BUFFER;
```

channel

Specifies the channel number of the channel to start the DA conversion. Valid channel numbers are 1 up to 8 for TIP550-10 and 1 up to 4 for TIP550-11.

correction

If this parameter is set to *TIP550_CORRECTION_ON* the driver performs an automatic offset and gain correction with factory calibration data stored in the TIP550 ID-PROM. If you do not need this correction set this parameter to *TIP550_CORRECTION_OFF.*

data

Specifies the analog output value in a range between –2048...2047 in bipolar mode (+/-10V), or in range between 0...4095 in unipolar mode (0V...10V).

Example

```
#include “tip550.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TIP550_IO_BUFFER ioBuf;

/*
**  Write a D/A value of 0x400 to channel 3, use data correction
*/
ioBuf.channel =    3;
ioBuf.correction = TIP550_CORRECTION_ON;
ioBuf.data =       0x400;

…
```

```
…

success = DeviceIoControl (
     hDevice,                // TIP550 handle
     IOCTL_TIP550_WRITE,
     &ioBuf,                 // parameter for the driver
     sizeof(ioBuf),
     NULL,                   // no data returned
     0,
     &NumBytes,              // size of returned Buffer
     0
);
if( success ) {
     printf(“Write successfully completed\n”);
}
else {
     ErrorHandler ("Device I/O control error”); // process error
}
```

Error Codes

All returned error codes are system error conditions.

3.1.3.2 IOCTL_TIP550_CONFIG

This control function configures the driver and the TIP550 to the hardware configured output mode.

The parameter *lpInBuffer* passes a pointer to an I/O buffer (*TIP550_CONF_BUFFER*) to the driver which contains parameter required to perform the operation.

lpOutBuffer must pass a *NULL* pointer to the device driver.

This function must be called before any output value is written.

The specified configuration must match to the jumper selected configuration of the device.

```
typedef struct {
        UCHAR           vrg1;
        UCHAR           vrg2;
} TIP550_CONF_BUFFER, *PTIP550_CONF_BUFFER;
```

vrg1

Specifies how the hardware is configured for channel 1...4. Valid values are:

TIP550_VRG_10_10	Specifies that the channels are configured in bipolar mode. Voltage range: -10V...+10V
TIP550_VRG_0_10	Specifies that the channels are configured in unipolar mode. Voltage range: 0V...+10V

vrg2

Specifies how the hardware is configured for channel 5...8. (This member is only valid for TIP550-10, and it will be ignored for TIP550-11). Valid values are:

TIP550_VRG_10_10	Specifies that the channels are configured in bipolar mode. Voltage range: -10V...+10V
TIP550_VRG_0_10	Specifies that the channels are configured in unipolar mode. Voltage range: 0V...+10V

Example

```
#include “tip550.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TIP550_CONF_BUFFER confBuf;

…
```

```
…

/*
**  Configure module (TIP550-10) for channel 1..4 in bipolar and
**  channel 5..8 in unipolar mode
*/
confBuf.vrg1 = TIP550_VRG_10_10;
confBuf.vrg2 = TIP550_VRG_0_10;

success = DeviceIoControl (
     hDevice,                    // TIP550 handle
     IOCTL_TIP550_CONFIG,
     &confBuf,                   // parameter for the driver
     sizeof(confBuf),
     NULL,
     0,
     &NumBytes,                  // size of returned Buffer
     0
);
if( success ) {
     printf("Configuration successful completed \n", RWBuf);
}
else {
     ErrorHandler ( "Device I/O control error" ); // process error
}
```

Error Codes

All returned error codes are system error conditions.