The Embedded I/O Company

TEWS TECHNOLOGIES

TIP570-SW-65

Windows 2000/XP Device Driver

16 Channel 12 Bit ADC / 8 Channel 12 Bit DAC

Version 1.0.x

User Manual

Issue 1.0.1

May 2010

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

TIP570-SW-65

Windows 2000/XP Device Driver

16 Channel 12 Bit ADC and
8 Channel 12 Bit DAC

Supported Modules:
TIP570-10
TIP570-11

Issue	Description	Date
1.0.0	First Issue	July 14, 2005
1.0.1	Carrier-Description added , Address of TEWS LLC removed	May 18, 2010

Table of Contents

1 INTRODUCTION 4
1.1 Device Driver 4
1.2 IPAC Carrier Driver 5
2 INSTALLATION 6
2.1 Software Installation 6
2.1.1 Windows 2000/XP 6
2.1.2 Confirming Installation 7
3 TIP570 DEVICE DRIVER PROGRAMMING 8
3.1 TIP570 Files and I/O Functions 8
3.1.1 Opening a TIP570 Device 8
3.1.2 Closing a TIP570 Device 10
3.1.3 TIP570 Device I/O Control Functions 11
3.1.3.1 IOCTL_TIP570_WRITE 13
3.1.3.2 IOCTL_TIP570_READ 15
3.1.3.3 IOCTL_TIP570_INFO 17

1 Introduction

1.1 Device Driver

The TIP570-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TIP570 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 TIP570 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 TIP570-SW-65 device driver includes the following functions:

- Getting A/D input value
- Setting D/A output value
- Use of factory programmed correction data for input and output
- Reading module information

The TIP570-SW-65 device driver supports the modules listed below:

TIP570-10	16 Channel 12-Bit ADC (Gain 1,2,5,10) 8 Channel 12-Bit DAC	(IPAC)
TIP570-11	16 Channel 12-Bit ADC (Gain 1,2,4,8) 8 Channel 12-Bit DAC	(IPAC)

In this document all supported modules and devices will be called TIP570. Specials for certain devices will be advised.

To get more information about the features and use of TIP570 devices it is recommended to read the manuals listed below.

TIP570 User manual

TIP570 Engineering Manual

CARRIER-SW-65 User 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 TIP570-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 in directory TIP570-SW-65 on the distribution media:

tip570.sys	Device driver binary
tip570.h	Header file with IOCTL code definitions
tip570.inf	Installation script
EmbeddedIoDeviceClass.dll	Windows WDM device class library
TIP570-SW-65-1.0.1.pdf	This document
example\tip570exa.c	example application
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Software Installation

The TIP570 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 TIP570 Device Driver on a Windows 2000/XP operating system.

After installing the TIP570 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 TIP570 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 distribution media.
   Click "***Next***" button to continue.
5. If a window shows up announcing that the Windows Logo Test has failed, click **"continue install"** to continue the installation.
6. Complete the device driver installation by clicking "***Finish***" to take all the changes effect.
7. Now copy all needed files (tip570.h, TIP570-SW-65.pdf) to the desired target directories.

After successful installation the TIP570 device driver will start immediately and creates devices (TIP570_1, TIP570_2, ...) for all recognized TIP570 modules.

2.1.2 Confirming 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 "***Embedded I/O*** ".
   The driver "***TEWS TECHNOLOGIES - TIP570 (DA/AD Converter)***" should appear.

3 TIP570 Device Driver Programming

The TIP570-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 TIP570 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TIP570 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TIP570 Device

Before you can perform any I/O the *TIP570* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TIP570* 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 TIP570 to open.
The *lpFileName* string should be of the form **\\.\TIP570_*x*** to open the device *x.* The ending x is a one-based number. The first device found by the driver is \\.\TIP570_1, the second \\.\TIP570_2 and so on.

dwDesiredAccess

Specifies the type of access to the TIP570. For the TIP570 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 TIP570, set to 0.

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TIP570 devices.

dwCreationDistribution

Specifies which action to take on files that exist and which action to take when files that do not exist. TIP570 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 TIP570 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TIP570 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call **GetLastError**.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TIP570_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TIP570 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 TIP570 Device

The **CloseHandle** function closes an open TIP570 handle.

```
BOOL CloseHandle(
      HANDLE hDevice;                          // handle to a TIP570 device to close
);
```

Parameters

hDevice

Identifies an open TIP570 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;

hDevice = CreateFile( … );

/* ... do some device I/O ...  */

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

See Also

CreateFile(), Win32 documentation CloseHandle()

3.1.3 TIP570 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 TIP570 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 *tip570.h*:

Value	Meaning
IOCTL_TIP570_WRITE	Set DAC output value
IOCTL_TIP570_READ	Get ADC input value
IOCTL_TIP570_INFO	Get module information

See behind for more detailed information on each control code.

To use these TIP570 specific control codes the header file tip570.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***.

See Also

Win32 documentation DeviceIoControl()

3.1.3.1 IOCTL_TIP570_WRITE

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

The parameter *lpInBuffer* passes a pointer to an I/O buffer (*T570_WRITE_BUFFER*) to the driver which contains parameter required to perform the operation like channel number, flags specifying the operation mode and the DA value to output.

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

```
typedef struct {
        int             chan;
        unsigned long   flags;
        short           data;
} T570_WRITE_BUFFER, *PT570_WRITE_BUFFER;
```

chan

This parameter specifies the channel number to be used for the DA conversion. Valid channel numbers are 1 up to 8.

flags

This parameter is an ORed set of a selection of the following flags defined in tip570.h:

define	**description**
T570_CORRECTION	If this flag is set the output data value will be corrected using the factory supplied correction data. If the flag is unset, the output data value will be used without correction.
T570_LATCHED	If this flag is set output value will be written into the DAC channel, but the output update will not be performed automatic. If the flag is unset the output will be written and the channel output will be updated. (For more information, refer to the TIP570 User Manual)
T570_LATCHED_OUTPUT	If this flag is set, all output lines will be updated. (For more information, refer to the TIP570 User Manual)

data

This parameter specifies the analog output value in a range between –2048...2047.

Example

```
#include “tip570.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TIP570_WRITE_BUFFER     wrBuf;

/*
**  Write a D/A value of 0x400 to channel 3, use data correction
*/
wrBuf.chan    =    3;
wrBuf.flags   =    T570_CORRECTION;
wrBuf.data    =    0x400;

success = DeviceIoControl (
     hDevice,               // TIP570 handle
     IOCTL_TIP570_WRITE,
     &wrBuf,                // parameter for the driver
     sizeof(wrBuf),
     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

ERROR_INSUFFICIENT_BUFFER	The input buffer is too small.
ERROR_INVALID_PARAMETER	Invalid flags or gain specified.
ERROR_MEMBER_NOT_IN_GROUP	Invalid channel specified.
ERROR_SEM_TIMEOUT	Timeout during conversion.

3.1.3.2 IOCTL_TIP570_READ

This control function starts an A/D conversion on the specified channel and returns the input value.

The parameter *lpInBuffer* passes a pointer to an I/O buffer (*T570_READ_BUFFER*) to the driver which contains parameter required to perform the operation like channel number and flags specifying the operation mode. *lpOutBuffer* also points to the I/O buffer, this will be used to store the input value.

```
typedef struct {
        int             chan;
        int             gain;
        unsigned long   flags;
        short           data;
} T570_READ_BUFFER, *PT570_READ_BUFFER;
```

chan

This parameter specifies the channel number to be used for the AD conversion. Valid channel numbers are 1 up to 16 for single-ended input and 1 up to 8 if differential input is used.

gain

This parameter specifies the gain that should be used for the conversion. Valid gains are 1, 2, 5 and 10 for TIP570-10. Valid values for TIP570-11 are 1, 2, 4 and 8.

flags

This parameter is an ORed set of a selection of the following flags defined in tip570.h:

define	**description**
T570_CORRECTION	If this flag is set the input data value will be corrected using the factory supplied correction data. If the flag is unset, the input data value will be returned without correction.
T570_DIFF	If this flag is set the channel will be used with a differential interface. If the flag is unset the channel will be used with a single-ended interface.
T570_PIPELINE	If this flag is set, the ADC will be used in pipeline mode. (For more information, refer to the TIP570 User Manual)

data

This parameter returns the analog input value. The returned data is in a range between –2048 and 2047.

Example

```
#include “tip570.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
TIP570_READ_BUFFER      rdBuf;

/*
**  Read differential input from channel 3, use data correction and gain 5
*/
wrBuf.chan    =    3;
wrBuf.gain    =    5;
wrBuf.flags   =    T570_DIFF | T570_CORRECTION;

success = DeviceIoControl (
     hDevice,               // TIP570 handle
     IOCTL_TIP570_READ,
     &rdBuf,                // parameter for the driver
     sizeof(rdBuf),
     &rdBuf,                // buffer filled by driver
     sizeof(rdBuf),
     &NumBytes,             // size of returned Buffer
     0
);

if( success ) {
     printf(“Read successfully completed\n”);
     printf(“  ADC value: %d\n”, rdBuf.data);
}
else {
     ErrorHandler ("Device I/O control error”);  // process error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The input buffer is too small.
ERROR_INVALID_PARAMETER	Invalid flags or gain specified.
ERROR_MEMBER_NOT_IN_GROUP	Invalid channel specified.
ERROR_NO_SYSTEM_RESOURCES	Error starting the conversion.

3.1.3.3 IOCTL_TIP570_INFO

This control function returns information about the TIP570 variant and the correction data for A/D and D/A conversions.

The parameter *lpOutBuffer* passes a pointer to an I/O buffer (*T570_INFO_BUFFER*) to the driver which will be used to return module information.

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

```
typedef struct {
    int             variant;
    signed char     ADC_offset_corr[4];
    signed char     ADC_gain_corr[4];
    signed char     DAC_offset_corr[8];
    signed char     DAC_gain_corr[8];
} T570_INFO_BUFFER, *PT570_INFO_BUFFER;
```

variant

The returned value specified the model variant of the TIP570. The function will return 10 if a TIP570-10 is in use and return 11 if a TIP570-11 is in use.

ADC_offset_corr[]

This array returns the factory stored correction data for offset correction of the A/D channels. The correction data depends on the gain. There for the array index is associated to the following gains.

index	Gain (TIP570-10)	Gain (TIP570-11)
0	1	1
1	2	2
2	5	4
3	10	8

ADC_gain_corr[]

This array returns the factory stored correction data for gain correction of the A/D channels. The correction data depends on the gain. There for the array index is associated to the following gains.

index	Gain (TIP570-10)	Gain (TIP570-11)
0	1	1
1	2	2
2	5	4
3	10	8

DAC_offset_corr[]

This array returns the factory stored correction data for offset correction of the D/A channels. The array index is associated with the channel number. The index is always (channel number – 1).

DAC_gain_corr[]

This array returns the factory stored correction data for gain correction of the D/A channels. The array index is associated with the channel number. The index is always (channel number – 1).

Example

```
#include “tip570.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
T570_INFO_BUFFER    infoBuf;

/*
**  Read module information data
*/
success = DeviceIoControl (
     hDevice,                // TIP570 handle
     IOCTL_TIP570_INFO,
     NULL,
     0,
     &infoBuf,               // buffer filles by driver
     sizeof(infoBuf),
     &NumBytes,              // size of returned Buffer
     0
);

if( success ) {
     printf(“Info read successfully completed\n”);
     printf(“  actual module is TIP570-%02d\n”, infoBuf.variant);
}
else {
     ErrorHandler ("Device I/O control error”);  // process error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The input buffer is too small.