The Embedded I/O Company

TEWS TECHNOLOGIES

TIP840-SW-65

Windows 2000/XP Device Driver

16 / 8 Channel 12-bit A/D

Version 1.0.x

User Manual

Issue 1.0.1

June 2008

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) 840 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TIP840-SW-65

Windows 2000/XP Device Driver

16 / 8 Channel 12-bit A/D

Supported Modules:
TIP840

Issue	Description	Date
1.0.0	First Issue	May 28, 2008
1.0.1	Files moved to subdirectory, Carrier Driver description added	June 23, 2008

Table of Contents

1 Introduction

1.1 Device Driver

The TIP840-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TIP840 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 TIP840 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 TIP840-SW-65 device driver supports the following features:

- Reading values from ADC channels with configurable gain and input interface
- Data correction with module specific correction data for ADC input values
- Reading module information (correction data and model type)

The TIP840-SW-65 device driver supports the modules listed below:

TIP840-10	8 single-ended channel 12-bit ADC (gain 1, 10, 100)	(IndustryPack)
TIP840-11	8 single-ended channel 12-bit ADC (gain 1, 2, 4, 8)	(IndustryPack)
TIP840-20	16 single-ended / 8 differential channel 12-bit ADC (gain 1, 10, 100)	(IndustryPack)
TIP840-21	16 single-ended / 8 differential channel 12-bit ADC (gain 1, 2, 4, 8)	(IndustryPack)

To get more information about the features and use of TIP840 devices it is recommended to read the manuals listed below.

TIP840 User manual

TIP840 Engineering Manual

CARRIER-SW-65 IPAC Carrier 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 TIP840-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 TIP840-SW-65 on the distribution media:

tip840.sys	Device driver binary
tip840.h	Header file with IOCTL code definitions
tip840.inf	Installation script
example\tip840exa.c	Example application
TIP840-SW-65-1.0.1.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

For installation the files have to be copied to the desired target directory.

2.1 Software Installation

The TIP840 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 TIP840 Device Driver on a Windows 2000/XP operating system.

After installing the TIP840 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 TIP840 driver distribution 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 diskette.
   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 (tip840.h, …) to the desired target directories.

After successful installation the TIP840 device driver will start immediately and creates devices (TIP840_1, TIP840_2, ...) for all recognized TIP840 modules.

2.1.2 Confirming Windows 2000/XP Installation

To confirm that the driver has been properly loaded in Windows 2000, perform the following steps:

1. From Windows 2000, 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 - TIP840 (16 / 8 Channel 12-bit ADC)***" should appear.

3 TIP840 Device Driver Programming

The TIP840-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 TIP840 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TIP840 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TIP840 Device

Before you can perform any I/O the TIP840 device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the TIP840 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 TIP840 to open.
The *lpFileName* string should be of the form **\\.\TIP840_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TIP840_1, the second \\.\TIP840_2 and so on.

dwDesiredAccess

Specifies the type of access to the TIP840. For the TIP840 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 TIP840, set to 0.

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TIP840 devices.

dwCreationDistribution

Specifies which action to take on files that exist and which action to take when files that do not exist. TIP840 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 TIP840 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TIP840 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call **GetLastError**.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TIP840_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TIP840 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 TIP840 Device

The **CloseHandle** function closes an open TIP840 handle.

BOOL CloseHandle(
    HANDLE *hDevice;* // handle to a TIP840 device to close
)

Parameters

hDevice

Identifies an open TIP840 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(
     “\\\\.\\TIP840_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TIP840 device always open existing
     0,                 // no overlapped I/O
     NULL
);
if(hDevice == INVALID_HANDLE_VALUE) {
 ErrorHandler("Could not open device"); // process error
}

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

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

See Also

CreateFile(), Win32 documentation CloseHandle()

3.1.3 TIP840 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 TIP840 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 *tip840.h*:

Value	Meaning
IOCTL_TIP840_ADCREAD	Read an ADC input value (Execute conversion)
IOCTL_TIP840_INFO	Read module specific information

See behind for more detailed information on each control code.

To use these TIP840 specific control codes the header file tip840.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_TIP840_ADCREAD

This read function starts an AD conversion at the specified input channel of the TIP840 and returns the converted analog value to the caller. A pointer to the read buffer structure (*TIP840_ADC_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.

```
typedef struct {
    int     chan;
    int     gain;
    int     flags;
    int     data;
} TIP840_ADC_READ_BUFFER;
```

chan

Specifies the channel number at which the conversion will be started. Valid channel numbers are 1...16 for single-ended and 1...8 for differential input (TIP840-20/-21), or 1…8 for single-ended input (TIP840-10/-11). Differential input is not supported by TIP840-10/-11.

gain

Specifies the gain for the input voltage amplifier. Valid gain values depend on the model type listed below:

Gain	valid for TIP840 variant
1	TIP840-10/-11/-20/-21
2	TIP840-11/-21
4	TIP840-11/-21
8	TIP840-11/-21
10	TIP840-10/-20
100	TIP840-10/-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).

Flag	Description
TIP840_ADC_DIFF	Use differential analog inputs. If this flag is omitted single-ended inputs will be selected
TIP840_CORRECTION	Perform an automatic offset and gain correction with factory calibration data stored in the TIP840 ID-PROM. If this flag is omitted the converted data will be read directly

data

Converted analog input value (two's complement).

Example

```
#include “tip840.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TIP840_ADC_READ_BUFFER rdBuf;

rdBuf.chan    = 1;
rdBuf.gain    = TIP840_GAIN_1;
rdBuf.flags   = TIP840_ADC_DIFF | TIP840_CORRECTION;

success = DeviceIoControl (
     hDevice,               // TIP840 handle
     IOCTL_TIP840_ADCREAD,
     &rdBuf,                // parameter for the driver
     sizeof(TIP840_ADC_READ_BUFFER),
     &rdBuf,                // data from the driver
     sizeof(TIP840_ADC_READ_BUFFER),
     &NumBytes,             // size of returned Buffer
     0);

if( success ) {
     printf( “ADC value: %d\n”, rdBuf.data );
} else {
     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_NO_SYSTEM_RESOURCES	An other conversion is already in progress

3.1.3.2 IOCTL_TIP840_INFO

This control function reads the module variant and the factory calibration data from the specified device and returns this information in the *TIP840_INFO_BUFFER* structure to the caller.

A pointer to the *TIP50_INFO_BUFFER* structure is passed by the argument *lpOutBuffer* to the driver. The argument *nOutBufferSize* specifies the length of this buffer. The pointer *lpInBuffer* must be NULL and *nInBufferSize* must be set to 0.

```
typedef struct {
    int         variant;
    int         adc_offset_corr[4];
    int         adc_gain_corr[4];
} TIP840_INFO_BUFFER;
```

variant

Returns the module variant.

Value	Module Variant
10	TIP840-10
11	TIP840-11
20	TIP840-20
21	TIP840-21

adc_offset_corr

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

adc_gain_corr

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

Index	Gain (TIP840-10/-20)	Gain (TIP840-11-/21)
0	1	1
1	10	2
2	100	4
3	not used	8

Example

```
#include “tip840.h”

HANDLE             hDevice;
BOOLEAN            success;
ULONG              NumBytes;
TIP840_INFO_BUFFER infoBuf;

success = DeviceIoControl (
     hDevice,               // TIP840 handle
     IOCTL_TIP840_INFO,
     NULL,                  // not used, set to NULL
     0,                     // not used, set to 0
     &infoBuf,
     sizeof(TIP840_INFO_BUFFER),
     &NumBytes,
     0);


if( success ) {
     printf( “Module Variant: TIP840-%d\n”, infoBuf.variant );
} else {
     ErrorHandler ( “Device I/O control error” ); // process error
}
```

Error Codes

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