The Embedded I/O Company

TEWS TECHNOLOGIES

TIP114-SW-65

Windows 2000/XP Device Driver

10 Channel Absolute Encoder Interface (SSI)

Version 1.0.x

User Manual

Issue 1.0

August 2004

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

TIP114-SW-65

10 Channel Absolute Encoder Interface (SSI)

Windows 2000/XP Device Driver

Issue	Description	Date
1.0	First Issue	August 27, 2004

Table of Contents

1 Introduction

The TIP114-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TIP114 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 TIP114 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 TIP114 device driver includes the following functions:

- Reading a single SSI channel (Non-Blocking)
- Configuring clock rate, packet size and coding

To understand all features of this device driver, it is recommended to read the TIP114 User Manual.

2 Installation

Following files are located on the distribution disk:

tip114.sys	Device driver binary
tip114.h	Header file with IOCTL code definitions and driver specific data types
tip114.inf	Installation script
TIP114-SW-65.pdf	This document
\Example\Example.c	Microsoft Visual C example application

2.1 Software Installation

The TIP114 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 TIP114 Device Driver on a Windows 2000/XP operating system.

After installing the TIP114 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 TIP114 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 (tip114.h, TIP114-SW-65.pdf) to the desired target directories.

After successful installation the TIP114 device driver will start immediately and creates devices (TIP114_1, TIP114_2, ...) for all recognized TIP114 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 "***TIP114***" should appear.

3 TIP114 Device Driver Programming

The TIP114-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 TIP114 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TIP114 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TIP114 Device

Before you can perform any I/O the *TIP114* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TIP114* 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 TIP114 to open.

The *lpFileName* string should be of the form **\\.\tip114_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\tip114_1, the second \\.\tip114_2 and so on.

dwDesiredAccess

Specifies the type of access to the TIP114. For the TIP114 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 TIP114, set to 0.

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TIP114 devices.

dwCreationDistribution

Specifies which action to take on files that exist and which action to take when files that do not exist. TIP114 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 TIP114 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TIP114 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call **GetLastError**.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\tip114_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TIP114 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 TIP114 Device

The **CloseHandle** function closes an open TIP114 handle.

```
BOOL CloseHandle(
      HANDLE hDevice;                          // handle to a TIP114 device to close
);
```

Parameters

hDevice

Identifies an already opened TIP114 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 TIP114 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 TIP114 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 *TIP114.h*:

Value	Meaning
IOCTL_TIP114_READ	Read a single SSI channel
IOCTL_TIP114_SET_CHAN_CONF	Set channel configuration of a single SSI channel
IOCTL_TIP114_GET_CHAN_CONF	Get channel configuration of a single SSI channel

See behind for more detailed information on each control code.

To use these TIP114 specific control codes the header file tip114.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_TIP114_READ

The read function starts an conversion at the specified input channel of the TIP114 and returns the encoder data to the caller. A pointer to the read buffer structure (*TIP114_IO_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 reading, you have to setup the specified channel using ioctl function IOCTL_TIP114_SET_CHAN_CONF and 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 *TIP114_IO_BUFFER* structure has the following layout:

```
typedef struct
{
      ULONG             chan;
      ULONG             value;
      ULONG             timeout;
} TIP114_IO_BUFFER;
```

chan

This parameter holds the channel of the specified tip114 module the driver should read from. Valid values are 0 to 9.

value

This parameter contains the channel value after the read call terminates.

timeout

Specifies the amount of time (in sec.) the caller is willing to wait for execution of read.

Example

```
#include "tip114.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TIP114_IO_BUFFER ioBuf;

ioBuf.chan =4;     /* read SSI channel 4 (starts by 0) */
ioBuf.timeout = 1; /* wait one second (max.) */
//
//  Send request to the device driver
//
success = DeviceIoControl (
     hCurrent,                  //  TIP114 handle
     IOCTL_TIP114_READ,         //  control code
     &ioBuf,
```

```
     sizeof(TIP114_IO_BUFFER),
     &ioBuf,
     sizeof(TIP114_IO_BUFFER),
     &NumBytes,                  //  number of bytes transferred
     NULL                        //  not over lapped
     );

//
//  Check the result of the last device I/O control operation
//
if( success ) {
     printf("\nRead successful, %d byte\n", NumBytes);
     printf( "\nChannel Value = %d (0x%x)\n", ioBuf.value, ioBuf.value);

}
else {
     /* Read failed -> Error handling */
}
```

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_NO_SYSTEM_RESOURCES	The selected channel is busy.
ERROR_SEM_TIMEOUT	Timeout occurred during conversion.

See Also

Win32 documentation DeviceIoControl(), TIP114 Hardware User Manual

3.1.3.2 IOCTL_TIP114_SET_CHAN_CONF

This control function writes the module configuration data of a certain TIP114 SSI channel.

A pointer to the *TIP114_CTRL_BUFFER* structure is passed by the argument *lpInBuffer* to the driver. The argument *nInBufferSize* specifies the length of this buffer.

The *TIP114_CTRL_BUFFER* structure has the following layout:

```
typedef struct
{
    ULONG       chan;
    ULONG       clock;
    ULONG       databits;
    ULONG       parity;
    ULONG       zerobit;
    ULONG       graycode;
    ULONG       timeout;
} TIP114_CTRL_BUFFER, *PTIP114_CTRL_BUFFER;
```

chan

This parameter holds the channel of the specified tip114 module the driver should setup. Valid values are 0 to 9.

clock

This parameter defines the clock rate of the specified channel. Valid values are 0 to 15. For more details see hardware manual.

databits

This parameter defines the data length of the specified channel. Valid values are 1 to 32. For more details see hardware manual.

parity

This parameter enables or disables parity checking. To disable parity checking set it to T114_NOP, otherwise T114_ODP for odd parity or T114_EVP for even parity. For more details see hardware manual.

zerobit

This parameter enables or disables parity with zerobit. To enable parity with zerobit set it to T114_ENABLE otherwise set it to T114_DISABLE. For more details see hardware manual.

graycode

This parameter enables or disables graycoded transmission. To enable it set this parameter to T114_ENABLE otherwise set it to T114_DISABLE.

timeout

Specifies the amount of time (in sec.) the caller is willing to wait for execution of ioctl.

EXAMPLE

```
#include “tip114.h”


HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
TIP114_CTRL_BUFFER  ctrlBuf;

/* Example values for HEIDENHAIN ROQ 424 SSI */
ctrlBuf.chan = 7;                /* channel 7 (starts by 0) */
ctrlBuf.clock = 1;          /* 1 microsec. clock cycle      */
ctrlBuf.databits = 25;           /* 25 databits per packet  */
ctrlBuf.parity = T114_NOP;       /* no parity               */
ctrlBuf.zerobit = T114_DISABLE;  /* no zero bit             */
ctrlBuf.graycode = T114_ENABLE;  /* gray coded data         */
ctrlBuf.timeout = 1;             /* conversion timeout 1 sec. */

success = DeviceIoControl (
     hDevice,               // TIP114 handle
     IOCTL_TIP114_SET_CHAN_CONF,
     &ctrlBuf,
     sizeof(ctrlBuf),
     NULL,                  // not used, set to NULL
     0,                     // not used, set to 0
     &NumBytes,
     0
);

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

Error Codes

ERROR_INSUFFICIENT_BUFFER	The input buffer is too small. Please check the argument *nInBufferSize.*
ERROR_MEMBER_NOT_IN_GROUP	Invalid channel number. Check parameter “chan”.

See Also

Win32 documentation DeviceIoControl(), TIP114 Hardware User Manual

3.1.3.3 IOCTL_TIP114_GET_CHAN_CONF

This control function reads the module configuration data of a certain TIP114 SSI channel.

A pointer to the *TIP114_CTRL_BUFFER* structure is passed by the argument *lpInBuffer* to the driver. The argument *nInBufferSize* specifies the length of this buffer.

The *TIP114_CTRL_BUFFER* structure has the following layout:

```
typedef struct
{
    ULONG       chan;
    ULONG       clock;
    ULONG       databits;
    ULONG       parity;
    ULONG       zerobit;
    ULONG       graycode;
    ULONG       timeout;
} TIP114_CTRL_BUFFER, *PTIP114_CTRL_BUFFER;
```

chan

This parameter holds the channel of the specified tip114 module the driver should retrieve. Valid values are 0 to 9.

clock

This parameter receives the clock rate of the specified channel. For more details see hardware manual.

databits

This parameter receives the data length of the specified channel. For more details see hardware manual.

parity

This parameter receives one of following parity checking states, T114_NOP for parity disabled, T114_ODP for odd parity or T114_EVP for even parity. For more details see hardware manual.

zerobit

This parameter receives one of following zerobit on/off-states, T114_ENABLE for parity with zerobit or T114_DISABLE for zerobit disabled. For more details see hardware manual.

graycode

This parameter receives one of following graycode on/off-state, T114_ENABLE for graycoded transmission or T114_DISABLE for graycode disabled.

timeout

Specifies the amount of time (in sec.) the caller is willing to wait for execution of ioctl.

EXAMPLE

```
#include “tip114.h”


HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
TIP114_CTRL_BUFFER  ctrlBuf;

ctrlBuf.chan = 0;                /* retrieves the 1st channel*/
ctrlBuf.timeout = 1;             /* timeout 1 sec. */

success = DeviceIoControl (
     hDevice,                // TIP114 handle
     IOCTL_TIP114_GET_CHAN_CONF,
     &ctrlBuf,
     sizeof(ctrlBuf),
     &ctrlBuf,
     sizeof(ctrlBuf),
     &NumBytes,
     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. Check parameter “chan”.

See Also

Win32 documentation DeviceIoControl(), TIP114 Hardware User Manual