The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC150-SW-65

Windows 2000/XP Device Driver

4, 3, 2 or 1 Channel Synchro/Resolver-to-Digital Converter

Version 1.0.x

User Manual

Issue 1.0.0

October 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

TPMC150-SW-65

Windows 2000/XP Device Driver

4, 3, 2 or 1 Channel Synchro/Resolver-to-Digital Converter

Supported Modules:
TPMC150

Issue	Description	Date
1.0.0	First Issue	October 1, 2010

Table of Contents

1 Introduction

The TPMC150-SW-65 Windows WDM (Windows Driver Model) device driver allows the operation of the TPMC150 PMC on an Intel or Intel-compatible x86 Windows 2000 or Windows XP operating system.

The standard file and device (I/O) functions (CreateFile, CloseHandle and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

The TPMC150-SW-65 device driver supports the following features:

- read converter data
- configure converter
- configure (synchron) read for multiple converters
- read encoder data
- configure encoder
- configure (synchron) read for multiple encoders
- read current state of digital input lines
- wait for digital input events

The TPMC150-SW-42 supports the modules listed below:

TPMC150-10	4 Channel Synchro/Resolver-to-Digital Converter	(PMC)
TPMC150-11	3 Channel Synchro/Resolver-to-Digital Converter	(PMC)
TPMC150-12	2 Channel Synchro/Resolver-to-Digital Converter	(PMC)
TPMC150-13	1 Channel Synchro/Resolver-to-Digital Converter	(PMC)

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

TPMC150 User Manual
TPMC150 Engineering Manual

2 Installation

Following files are located in directory TPMC150-SW-65 on the distribution media:

tpmc150.sys	Windows WDM driver binary
tpcm150.inf	Windows WDM installation script
tpmc150.h	Header file with IOCTL codes and structure definitions
EmbeddedIoDeviceClass.dll	Windows WDM device class library
tpmc150-SW-65-1.0.0.pdf	This document
api\tpmc150api.h	API include file
api\tpmc150api.c	API source file
example\tpmc150exa.c	Example application
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TPMC150-SW-65 Device Driver on a Windows 2000 / XP operating system.

After installing the hardware 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. In Drive A, insert the driver disk; select "***Disk Drive***" 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. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Repeat the steps above for each found module of the TPMC150 product family.
7. Copy needed files (tpmc150.h, API files) to desired target directory.

After successful installation a device is created for each found module (TPMC150_1, TPMC150_2 ...).

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 "***Embedded I/O***".
   The driver "***TEWS TECHNOLOGIES TPMC150 (Synchro/Resolver-to-Digital Converter)***" should appear for each installed device.

3 Device Driver Programming

The TPMC150-SW-65 Windows WDM device driver is a kernel mode device driver using Direct I/O.

The standard file and device (I/O) functions (CreateFile, CloseHandle and DeviceIoControl) provide the basic interface for opening and closing a resource 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.)

4 API Documentation

4.1 General Functions

4.1.1 tpmc150Open()

NAME

tpmc150Open() – opens a device.

SYNOPSIS

```
TPMC150_DEV tpmc150Open
(
      char          *DeviceName
);
```

DESCRIPTION

Before I/O can be performed to a device, a file descriptor must be opened by a call to this function.

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV   FileDescriptor;

/*
** open file descriptor to device
*/
FileDescriptor = tpmc150Open(“\\\\.\\TPMC150_1”);
if (FileDescriptor < 0)
{
     /* handle open error */
}
```

RETURNS

A device descriptor number, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system.

4.1.2 tpmc150Close()

NAME

tpmc150Close() – closes a device.

SYNOPSIS

```
int tpmc150Close
(
      TPMC150_DEV            FileDescriptor
);
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

FileDescriptor

This value specifies the file descriptor to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        FileDescriptor;
int                result;

/*
** close file descriptor to device
*/
result = tpmc150Close(FileDescriptor);
if (result < 0)
{
     /* handle close error */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system.

4.1.3 tpmc150ReadConverter

NAME

tpmc150ReadConverter – read value from converter channel

SYNOPSIS

```
int tpmc150ReadConverter
(
    TPMC150_DEV         pDev,
    unsigned char       chanNo,
    unsigned short      *pValue,
    uint32_t            *pState
);
```

DESCRIPTION

This function reads the value from the specified converter channel. Depending on the configuration the function will return the current value of the converter or a value latched by a read of another converter channel.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the converter channel. Allowed values are 1 up to the number of available converter channels on the specified device.

pValue

This parameter points to a buffer, where the function will store the read converter value.

pState

This parameter points to a buffer, where the function will store the status data of the converter. The following flags may be set:

Flag	Description
TPMC150_IO_F_CONDATA_BIT	If this bit is set the Built-in-Self-Test failed
TPMC150_IO_F_CONDATA_NOADAMOUNT	If this bit is set there is no adapter mounted
TPMC150_IO_F_CONDATA_MOTDIR	This bit indicates the direction of the motion. 0: down (clockwise, angle decreasing) 1: up (counter clockwise, angle increasing)

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;
unsigned short     convVal;
uint32_t           convState;

/*
** read current value and state of converter channel 3
*/
result = tpmc150ReadConverter(pDev,
                              3,
                              &convVal,
                              &convState);
if (result < 0)
{
     /* handle error */
}
else
{
     printf(“converter #3:\n”);
     printf(“    value: 0x%04X\n”, convVal);
     printf(“        Direction: %s\n”,
          (convState & TPMC150_IO_F_CONDATA_MOTDIR) ? “up” : “down”);
     if (convState & TPMC150_IO_F_CONDATA_BIT)
          printf(“        Built_in_Self_Test failed\n”);
     if (convState & TPMC150_IO_F_CONDATA_NOADAMOUNT)
          printf(“        No adapter mounted\n”);
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	A NULL pointer is referenced for an input value or unknown channel specified

4.1.4 tpmc150ConfigureConverter

NAME

tpmc150ConfigureConverter – configure converter channel

SYNOPSIS

```
int tpmc150ConfigureConverter
(
    TPMC150_DEV             pDev,
    unsigned char           chanNo,
    int                     resolution,
    BOOLEAN                 syncStatLatchEna,
    BOOLEAN                 syncConversionEna
);
```

DESCRIPTION

This function configures the specified converter channel.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the converter channel. Allowed values are 1 up to the number of available converter channels on the specified device.

resolution

The argument specifies the resolution in bits. Allowed resolutions are 10, 12, 14 and 16 bit.

syncStatLatchEna

This argument specifies if the synchronous status latch shall be enabled (TRUE) or disabled (FALSE).

syncConversionEna

This argument specifies if the synchronous conversion shall be enabled (TRUE) or not (FALSE). This parameter will not be used for channel 2 and 4.
(If channel 1 is specified, channel 1 and 2 will be coupled - If channel 3 is specified, channel 3 and 4 will be coupled.)

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;

/*
** configure converter channel 2
**   - resolution: 16bit
**   - no synchronization
*/
result = tpmc150ConfigureConverter(pDev,
                                   2,
                                   16,
                                   FALSE,
                                   FALSE);
if (result < 0)
{
     /* handle error */
}
else
{
     /* configuration succeeded */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	An invalid parameter value has been specified or unknown channel specified

4.1.5 tpmc150ConfigMultiConvRead

NAME

tpmc150ConfigMultiConvRead – configure (synchronous) multiple converter read

SYNOPSIS

```
int tpmc150ConfigMultiConvRead
(
    TPMC150_DEV         pDev,
    BOOLEAN             multiReadEna,
    BOOLEAN             multiChanEna1,
    BOOLEAN             multiChanEna2,
    BOOLEAN             multiChanEna3,
    BOOLEAN             multiChanEna4
);
```

DESCRIPTION

This function configures the multiple converter read. The converter channels configured for multiple read will be synchronized. The first read to a multiple read enabled converter channel latches the value/state of the other enabled converter channels.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

multiReadEna

This argument specifies if multi read shall be enabled (TRUE) or disabled (FALSE).

multiChanEna1

This argument specifies if converter channel 1 shall be connected to the multiple read (TRUE) or be used unsynchronized.

multiChanEna2

This argument specifies if converter channel 2 shall be connected to the multiple read (TRUE) or be used unsynchronized.

multiChanEna3

This argument specifies if converter channel 3 shall be connected to the multiple read (TRUE) or be used unsynchronized.

multiChanEna4

This argument specifies if converter channel 4 shall be connected to the multiple read (TRUE) or be used unsynchronized.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;

/*
** couple converter channel 1, 2 and 4 for synchronous read
*/
result = tpmc150ConfigMultiConvRead(pDev,
                                    TRUE,
                                    TRUE, TRUE, FALSE, TRUE);
if (result < 0)
{
     /* handle error */
}
else
{
     /* multiple read configured */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system.

4.1.6 tpmc150ReadEncoder

NAME

tpmc150ReadEncoder – read value from an encoder channel

SYNOPSIS

```
int tpmc150ReadEncoder
(
    TPMC150_DEV         pDev,
    unsigned char       chanNo,
    uint32_t            *pValue
);
```

DESCRIPTION

This function reads the value from the specified encoder channel. Depending on the configuration the function will return the current value of the encoder or a value latched by a read of another encoder channel.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the encoder channel. Allowed values are 1 up to the number of available encoder channels on the specified device.

pValue

This parameter points to a buffer, where the function will store the read encoder value.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;
uint32_t           encVal;

…
```

…

```
/*
** read current value of encoder channel 1
*/
result = tpmc150ReadEncoder(pDev,
                            1,
                            &encValue);
if (result < 0)
{
     /* handle error */
}
else
{
     printf("value: 0x%08X\n", encValue);
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	A NULL pointer is referenced for an input value or unknown channel specified

4.1.7 tpmc150WriteEncoderPreload

NAME

tpmc150WriteEncoderPreload – set the preload value of an encoder channel

SYNOPSIS

```
int tpmc150WriteEncoderPreload
(
        TPMC150_DEV             pDev,
        unsigned char           chanNo,
        uint32_t                value,
        BOOLEAN                 immediateLoad
);
```

DESCRIPTION

This function sets the preload value for the specified encoder channel.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the encoder channel. Allowed values are 1 up to the number of available encoder channels on the specified device.

value

This argument specifies the new preload value.

immediateLoad

This argument specifies if an immediate load shall be executed (TRUE), or if the value shall just be used if a ‘normal’ load event occurs (FALSE).

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;

/*
** set preload value of channel 4 to 0 and immediately load the value
*/
result = tpmc150WriteEncoderPreload(pDev,
                                    4,
                                    0,
                                    TRUE);
if (result < 0)
{
     /* handle error */
}
else
{
     /* preload value set properly */;
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	Unknown channel specified

4.1.8 tpmc150ConfigureEncoder

NAME

tpmc150ConfigureEncoder – configure encoder channel

SYNOPSIS

```
int tpmc150ConfigureEncoder
(
        TPMC150_DEV             pDev,
        unsigned char           chanNo,
        unsigned char           signalMode,
        unsigned char           referenceMode,
        BOOLEAN                 incEncEmulationEna,
        BOOLEAN                 incEncEmulationOutputEna
);
```

DESCRIPTION

This function configures the specified converter channel.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the encoder channel. Allowed values are 1 up to the number of encoder channels on the specified device.

signalMode

The argument specifies the signal mode. Valid signal modes are:

Signal mode	Description
TPMC150_IO_M_CONESIGANA_OFF	disable counter
TPMC150_IO_M_CONESIGANA_1X	1x - single
TPMC150_IO_M_CONESIGANA_2X	2x - double
TPMC150_IO_M_CONESIGANA_4X	4x - quad

referenceMode

The argument specifies the reference mode. Valid reference modes are:

Signal mode	Description
TPMC150_IO_M_CONEREF_NONE	None reference mode
TPMC150_IO_M_CONEREF_REF	Reference mode
TPMC150_IO_M_CONEREF_AUTOREF	Auto reference mode
TPMC150_IO_M_CONEREF_INDEX	Index mode

incEncEmulationEna

The argument specifies if the incremental encoder emulation shall be enable (TRUE) or disabled (FALSE).

incEncEmulationOutputEna

The argument specifies if the incremental encoder emulation output signal shall be enable (TRUE) or disabled (FALSE).

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;

/*
** configure encoder channel 2
**    4x – quad mode
**    none reference mode
**    encoder emulation and output enabled
*/
result = tpmc150ConfigureEncoder(pDev,
                                 2,
                                 TPMC150_IO_M_CONESIGANA_4X,
                                 TPMC150_IO_M_CONEREF_NONE,
                                 TRUE,
                                 TRUE);
if (result < 0)
{
     /* handle error */
}
else
{
     /* configuration succeeded */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	An invalid parameter value has been specified or unknown channel specified

4.1.9 tpmc150ConfigMultiEncRead

NAME

tpmc150ConfigMultiEncRead – configure (synchronous) multiple encoder read

SYNOPSIS

```
int tpmc150ConfigMultiEncRead
(
    TPMC150_DEV         pDev,
    BOOLEAN             multiReadEna,
    BOOLEAN             multiChanEna1,
    BOOLEAN             multiChanEna2,
    BOOLEAN             multiChanEna3,
    BOOLEAN             multiChanEna4
);
```

DESCRIPTION

This function configures the multiple encoder read. The encoder channels configured for multiple read will be synchronized. The first read to a multiple read enabled encoder channel latches the value/state of the other enabled encoder channels.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

multiReadEna

This argument specifies if multi read shall be enabled (TRUE) or disabled (FALSE).

multiChanEna1

This argument specifies if encoder channel 1 shall be connected to the multiple read (TRUE) or be used unsynchronized.

multiChanEna2

This argument specifies if encoder channel 2 shall be connected to the multiple read (TRUE) or be used unsynchronized.

multiChanEna3

This argument specifies if encoder channel 3 shall be connected to the multiple read (TRUE) or be used unsynchronized.

multiChanEna4

This argument specifies if encoder channel 4 shall be connected to the multiple read (TRUE) or be used unsynchronized.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;
uint32_t           in_value;

/*
** couple encoder channel 1, 2 and 4 for synchronous read
*/

result = tpmc150ConfigMultiEncRead(pDev,
                                   TRUE,
                                   TRUE, TRUE, FALSE, TRUE);
if (result < 0)
{
     /* handle error */
}
else
{
     /* multiple read configured */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system.

4.1.10 tpmc150GetDigitialInput

NAME

tpmc150GetDigitialInput – read state of the digital input lines

SYNOPSIS

```
int tpmc150GetDigitialInput
(
      TPMC150_DEV              pDev,
      unsigned char            *pData
);
```

DESCRIPTION

This function reads the current state of the digital input lines.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

pData

This parameter points to a buffer where the function will store the digital input states. A set bit specifies an active and a reset bit a passive input state.
Bit 0 specifies the state of digital input 1, bit 1 the state of digital input 2, bit 2 the state of digital input 3, and bit 3 the state of digital input 4.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;
unsigned char      diginVal;

…
```

…

```
/*
** read digital input state
*/
result = tpmc150GetDigitialInput(pDev, &diginVal);
if (result < 0)
{
     /* handle error */
}
else
{
     printf("value: 0x%02X\n", diginVal);
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	A NULL pointer is referenced for an input value

4.1.11 tpmc150WaitHighTrans

NAME

tpmc150WaitHighTrans – wait for a high transition event on digital input line

SYNOPSIS

```
int tpmc150WaitHighTrans
(
    TPMC150_DEV         pDev,
    int                 chanNo,
    int                 msTimeout
);
```

DESCRIPTION

This function waits until a low-to-high transition on a specified digital input channel occurs.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the digital input channel where the event shall occur. Allowed values are 1 up to 4.

msTimeout

This argument specifies the time (in ms) the function is willing to wait for the event. If the specified time has passed and the event has not occurred, the function will return with an appropriate error code. This function will never timeout if a value of -1 is specified.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;

/*
** wait for high transition on channel 1, timeout after 10 seconds
*/
result = tpmc150WaitHighTrans(pDev,
                              1,
                              10000);
if (result < 0)
{
     /* handle error */
}
else
{
     /* event has occurred */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	Unknown channel specified

4.1.12 tpmc150WaitLowTrans

NAME

tpmc150WaitLowTrans – wait for a low transition event on digital input line

SYNOPSIS

```
int tpmc150WaitLowTrans
(
    TPMC150_DEV         pDev,
    int                 chanNo,
    int                 msTimeout
);
```

DESCRIPTION

This function waits until a high-to-low transition on a specified digital input channel occurs.

PARAMETERS

pDev

This parameter specifies the device descriptor to the hardware module retrieved by a call to the corresponding open-function.

chanNo

This parameter specifies the digital input channel where the event shall occur. Allowed values are 1 up to 4.

msTimeout

This argument specifies the time (in ms) the function is willing to wait for the event. If the specified time has passed and the event has not occurred, the function will return with an appropriate error code. This function will never timeout if a value of -1 is specified.

EXAMPLE

```
#include “tpmc150api.h”

TPMC150_DEV        pDev;
int                result;
uint32_t           in_value;

/*
** wait for low transition on channel 3, timeout after 12 seconds
*/
result = tpmc150WaitLowTrans(pDev,
                             3,
                             12000);
if (result < 0)
{
     /* handle error */
}
else
{
     /* event has occurred */
}
```

RETURNS

Zero, or -1 if the function fails. An error code will be stored in *errno*.

ERROR CODES

To get extended error information, call GetLastError.

The error code is a standard error code set by the I/O system or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ERROR_INVALID_PARAMETER	Unknown channel specified