The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC150-SW-65

Windows Device Driver

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

Version 2.0.x

User Manual

Issue 2.0.0

December 2013

powerBridge Computer

Ehlbeek 15a
30938 Burgwedel
fon 05139-9980-0
fax 05139-9980-49

www.powerbridge.de
info@powerbridge.de

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7 25469 Halstenbek, Germany
49 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19
ail: info@tews.com www.tews.com

TPMC150-SW-65

Windows 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
2.0.0	API modified	December 2, 2013

Table of Contents

1 Introduction

The TPMC150-SW-65 Windows device driver is a kernel mode driver which allows the operation of supported hardware modules on an Intel or Intel-compatible Windows operating system. Supported Windows versions are:

- Windows 2000
- Windows XP
- Windows XP Embedded
- Windows 7 (32bit and 64bit)

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-65 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

2 Installation

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

i386\	Directory containing driver files for 32bit Windows versions
amd64\	Directory containing driver files for 64bit Windows versions
installer_32bit.exe	Installation tool for 32bit systems (Windows XP or later)
installer_64bit.exe	Installation tool for 64bit systems (Windows XP or later)
tpmc150.inf	Windows installation script
tpmc150.h	Header file with IOCTL codes and structure definitions
example\tpmc150exa.c	Example application
api\tpmc150api.c	Application Programming Interface source
api\tpmc150api.h	Application Programming Interface header
TPMC150-SW-65-2.0.0.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000

This section describes how to install the TPMC150 Device Driver on a Windows 2000 operating system.

After installing the TPMC150 card(s) and boot-up your system, Windows 2000 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 TPMC1580 driver media; 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 media.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Now copy all needed files (tpmc150.h and API files) to the desired target directories.

After successful installation the TPMC150 device driver will start immediately and creates devices (TPMC150_1, TPMC150_2 ...) for all recognized TPMC150 modules.

2.1.2 Windows 7 / XP

This section describes how to install the TPMC150-SW-65 Device Driver on a Windows 7 (32bit or 64bit) or Windows XP (32-bit) operating system.

Depending on the operating system type, execute the installer binaries for either 32bit or 64bit systems. This will install all required driver files using an installation wizard.

Copy needed files (tpmc150.h and API files) to desired target directory.

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

2.2 Confirming Driver Installation

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

1. Open the Windows Device Manager:
    a. For Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***" and click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
    b. For Windows 7, open the "***Control Panel***" from "***My Computer***" and then click the "***Device Manager***" entry.
2. 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 API Documentation

3.1 General Functions

3.1.1 tpmc150Open

NAME

tpmc150Open – opens a device.

SYNOPSIS

```
TPMC150_HANDLE tpmc150Open
(
        char            *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle 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. The device naming depends on the used operating system, and is shown in the following table:

Device Number	Device Name
0	\\\\.\\TPMC150_1
1	\\\\.\\TPMC150_2

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;

/*
** open the specified device (Windows Example)
*/
hdl = tpmc150Open("\\\\.\\TPMC150_1");
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. An error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno.*

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

3.1.2 tpmc150Close

NAME

tpmc150Close – closes a device.

SYNOPSIS

```
TPMC150_STATUS tpmc150Close
(
      TPMC150_HANDLE        hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE      hdl;
TPMC150_STATUS      result;

/*
** close the device
*/
result = tpmc150Close(hdl);
if (result != TPMC150_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid

3.2 Device Access Functions

3.2.1 tpmc150ReadConverter

NAME

tpmc150ReadConverter – read value from converter channel

SYNOPSIS

```
TPMC150_STATUS tpmc150ReadConverter
(
    TPMC150_HANDLE    hdl,
    int               chanNo,
    unsigned short    *pValue,
    unsigned int      *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

hdl

This value specifies the device handle 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_HANDLE     hdl;
TPMC150_STATUS     result;
unsigned short     convVal;
unsigned int       convState;

/*
** read current value and state of converter channel 3
*/
result = tpmc150ReadConverter(hdl,
                              3,
                              &convVal,
                              &convState);
if (result != TPMC150_OK)
{
     /* 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");
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified or NULL pointer referenced

3.2.2 tpmc150ConfigureConverter

NAME

tpmc150ConfigureConverter – configure converter channel

SYNOPSIS

```
TPMC150_STATUS tpmc150ConfigureConverter
(
    TPMC150_HANDLE      hdl,
    int                 chanNo,
    int                 resolution,
    unsigned int        flags
)
```

DESCRIPTION

This function configures the specified converter channel.

PARAMETERS

hdl

This value specifies the device handle 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.

flags

This argument specifies a set of bit flags for configuration:

Value	Description
TPMC150_LATCH_ENABLE	If this flag is set, synchronous status latch will be enabled.
TPMC150_CONV_ENABLE	If this flag is set, synchronous conversion will be enabled. If the flag is set for channel 1, channel 1 and 2 will be coupled. If the flag is set for channel 3, channel 3 and 4 will be coupled. This flag will be ignored for channel 2 and 4.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;
/*
** configure converter channel 2
**   - resolution: 16bit
**   - no synchronization
*/
result = tpmc150ConfigureConverter(hdl,
                                   2,
                                   16,
                                   0);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     /* configuration succeeded */
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified or NULL pointer referenced.

3.2.3 tpmc150ConfigMultiConvRead

NAME

tpmc150ConfigMultiConvRead – configure (synchronous) multiple converter read

SYNOPSIS

```
TPMC150_STATUS tpmc150ConfigMultiConvRead
(
    TPMC150_HANDLE      hdl,
    unsigned int        channels,
    unsigned int        flags
)
```

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

hdl

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

channels

This argument specifies the converter channels connected to the multichannel read mode. This value receives a bit-field of the desired channels. Bit 0 corresponds to channel 1, bit 1 corresponds to channel 2 and so on. Using the macro TPMC150_CHAN_ENABLE(*chanNo*) returns the corresponding bit to enable channel *chanNo*. Possible values for *chanNo* are 1 up to the number of available converter channels on the specified device.

flags

This argument specifies a set of bit flags that controls the configuration:

Value	Description
TPMC150_MULTIREAD_ENABLE	If this flag is set, multi read will be enabled. If this flag is not set, multi read is disabled.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;

/*
** couple converter channel 1, 2 and 4 for synchronous read
*/
result = tpmc150ConfigMultiConvRead(hdl,
                                    TPMC150_CHAN_ENABLE(1) |
                                    TPMC150_CHAN_ENABLE(2) |
                                    TPMC150_CHAN_ENABLE(4),
                                    TPMC150_MULTIREAD_ENABLE);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     /* multiple read configured */
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.2.4 tpmc150ReadEncoder

NAME

tpmc150ReadEncoder – read value from an encoder channel

SYNOPSIS

```
TPMC150_STATUS tpmc150ReadEncoder
(
    TPMC150_HANDLE      hdl,
    int                 chanNo,
    unsigned int        *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

hdl

This value specifies the device handle 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_HANDLE     hdl;
TPMC150_STATUS     result;
unsigned int       encValue;

/*
** read current value of encoder channel 1
*/
result = tpmc150ReadEncoder(hdl,
                            1,
                            &encValue);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     printf("value: 0x%08X\n", encValue);
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified or NULL pointer referenced.

3.2.5 tpmc150WriteEncoderPreload

NAME

tpmc150WriteEncoderPreload – set the preload value of an encoder channel

SYNOPSIS

```
TPMC150_STATUS tpmc150WriteEncoderPreload
(
    TPMC150_HANDLE      hdl,
    int                 chanNo,
    unsigned int        value,
    unsigned int        flags
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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.

flags

This argument specifies a set of bit flags for configuration:

Value	Description
TPMC150_IMMEDIATE_LOAD	If this flag is set, an immediate load will be executed and the new value will be loaded.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;

/*
** set preload value of channel 4 to 0 and immediately load the value
*/
result = tpmc150WriteEncoderPreload(hdl,
                                    4,
                                    0,
                                    TPMC150_IMMEDIATE_LOAD);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     /* preload values set*/;
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified.

3.2.6 tpmc150ConfigureEncoder

NAME

tpmc150ConfigureEncoder – configure encoder channel

SYNOPSIS

```
TPMC150_STATUS tpmc150ConfigureEncoder
(
    TPMC150_HANDLE      hdl,
    int                 chanNo,
    unsigned int        signalMode,
    unsigned int        referenceMode,
    unsigned int        flags
)
```

DESCRIPTION

This function configures the specified converter channel.

PARAMETERS

hdl

This value specifies the device handle 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:

Reference 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

flags

This argument specifies a set of bit flags for configuration:

Value	Description
TPMC150_ENCEMU_ENABLE	If this flag is set, the incremental encoder emulation will be enabled.
TPMC150_ENCEMUOUT_ENABLE	If this flag is set, the incremental encoder emulation output signal will be enabled.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;

/*
** configure encoder channel 2
**    4x - quad mode
**    none reference mode
**    encoder emulation and output enabled
*/
result = tpmc150ConfigureEncoder(hdl,
                                 2,
                                 TPMC150_IO_M_CONESIGANA_4X,
                                 TPMC150_IO_M_CONEREF_NONE,
                                 (TPMC150_ENCEMU_ENABLE |
                                  TPMC150_ENCEMUOUT_ENABLE)
                                 );
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     /* configuration succeeded */
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified.

3.2.7 tpmc150ConfigMultiEncRead

NAME

tpmc150ConfigMultiEncRead – configure (synchronous) multiple encoder read

SYNOPSIS

```
TPMC150_STATUS tpmc150ConfigMultiEncRead
(
    TPMC150_HANDLE      hdl,
    unsigned int        channels,
    unsigned int        flags
)
```

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

hdl

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

channels

This argument specifies the encoder channels connected to the multichannel read mode. This value receives a bit-field of the desired channels. Bit 0 corresponds to channel 1, bit 1 corresponds to channel 2 and so on. Using the macro TPMC150_CHAN_ENABLE(*chanNo*) returns the corresponding bit to enable channel *chanNo*. Possible values for *chanNo* are 1 up to the number of available encoder channels on the specified device.

flags

This argument specifies a set of bit flags for configuration:

Value	Description
TPMC150_MULTIREAD_ENABLE	If this flag is set, multi read will be enabled. If this flag is not set, multi read is disabled.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;
unsigned int       in_value;

/*
** couple encoder channel 1, 2 and 4 for synchronous read
*/
result = tpmc150ConfigMultiEncRead(hdl,
                                   TPMC150_CHAN_ENABLE(1) |
                                   TPMC150_CHAN_ENABLE(2) |
                                   TPMC150_CHAN_ENABLE(4),
                                   TPMC150_MULTIREAD_ENABLE);
if (result != TPMC150_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.2.8 tpmc150GetDigitialInput

NAME

tpmc150GetDigitialInput – read state of the digital input lines

SYNOPSIS

```
TPMC150_STATUS tpmc150GetDigitialInput
(
    TPMC150_HANDLE      hdl,
    unsigned int        *pData
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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 on 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_HANDLE     hdl;
TPMC150_STATUS     result;
unsigned int       diginVal;

/*
** read digital input state
*/
result = tpmc150GetDigitialInput(hdl, &diginVal);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     printf("value: 0x%02X\n", diginVal);
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified or NULL pointer referenced.

3.2.9 tpmc150WaitHighTrans

NAME

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

SYNOPSIS

```
TPMC150_STATUS tpmc150WaitHighTrans
(
    TPMC150_HANDLE      hdl,
    int                 chanNo,
    int                 msTimeout
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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. For Windows systems, the timeout granularity is in seconds. If the specified time has passed and the event has not occurred, the function will return with an appropriate error code. The function will never timeout if a value of -1 is specified.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;

/*
** wait for high transition on channel 1, timeout after 10 seconds
*/
result = tpmc150WaitHighTrans(hdl, 1, 10000);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     /* event has occurred */
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified.
TPMC150_ERR_TIMEDOUT	Waiting for event has exceeded specified time.

3.2.10 tpmc150WaitLowTrans

NAME

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

SYNOPSIS

```
TPMC150_STATUS tpmc150WaitLowTrans
(
    TPMC150_HANDLE      hdl,
    int                 chanNo,
    int                 msTimeout
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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. For Windows systems, the timeout granularity is in seconds. If the specified time has passed and the event has not occurred, the function will return with an appropriate error code. The function will never timeout if a value of -1 is specified.

EXAMPLE

```
#include "tpmc150api.h"

TPMC150_HANDLE     hdl;
TPMC150_STATUS     result;

/*
** wait for low transition on channel 1, timeout after 10 seconds
*/
result = tpmc150WaitLowTrans(hdl, 1, 10000);
if (result != TPMC150_OK)
{
     /* handle error */
}
else
{
     /* event has occurred */
}
```

RETURN VALUE

On success, TPMC150_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC150_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC150_ERR_INVAL	Invalid input argument specified.
TPMC150_ERR_TIMEDOUT	Waiting for event has exceeded specified time.