The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC500-SW-65

Windows Device Driver

Optically Isolated 32 Channel 12 Bit ADC

Version 2.0.x

User Manual

Issue 2.0.0

December 2013

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

TPMC500-SW-65

Windows Device Driver

Optically Isolated 32 Channel 12 Bit ADC

Supported Modules:
TPMC500

Issue	Description	Date
1.0	First Issue	May 12, 2004
1.1	Errors corrected	August 5, 2004
1.1.1	Release information added	May 11, 2005
2.0.0	Windows 7 support, API functions added	December 9, 2013

Table of Contents

1 Introduction

The TPMC500-SW-65 Windows device driver is a kernel mode driver which allows the operation of the supported hardware module 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 TPMC500-SW-65 device driver supports the following features:

- reading converted AD values from a specified channel
- configuring the sequencer for a free running measurement
- direct transfer of converted AD values to a dynamic ring buffer in the user space of the application task (Direct I/O)
- AD data correction with factory calibration data stored in the onboard EEPROM

The TPMC500-SW-65 device driver supports the modules listed below:

TPMC500	Optically Isolated 32 Channel 12 Bit ADC	(PMC)

To get more information about the features and use of TPMC500 devices it is recommended to read the manuals listed below.

TPMC500 User Manual

2 Installation

Following files are located in directory TPMC500-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)
tpmc500.inf	Windows installation script
tpmc500.h	Header file with IOCTL codes and structure definitions
example\tpmc500exa.c	Example application
api\tpmc500api.c	Application Programming Interface source
api\tpmc500api.h	Application Programming Interface header
TPMC500-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 TPMC500 Device Driver on a Windows 2000 operating system.

After installing the TPMC500 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 TPMC500 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 (tpmc500.h and API files) to the desired target directories.

After successful installation the TPMC500 device driver will start immediately and creates devices (TPMC500_1, TPMC500_2 ...) for all recognized TPMC500 modules.

2.1.2 Windows 7 / XP

This section describes how to install the TPMC500-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 (tpmc500.h and API files) to desired target directory.

After successful installation a device is created for each module found (TPMC500_1, TPMC500_2 ...).

2.1.3 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 - TPMC500 32(16) Channel 12-Bit ADC (TPMC500)***" should appear for each installed device.

3 API Documentation

3.1 General Functions

3.1.1 tpmc500Open

NAME

tpmc500Open – Opens a Device

SYNOPSIS

```
TPMC500_HANDLE tpmc500Open
(
        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 “tpmc500api.h”

TPMC500_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tpmc500Open(“\\\\.\\TPMC500_1” );
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. To get extended error information, call ***GetLastError***.

ERROR CODES

All error codes are standard error codes set by the I/O system.

3.1.2 tpmc500Close

NAME

tpmc500Close – Closes a Device

SYNOPSIS

```
TPMC500_STATUS tpmc500Close
(
      TPMC500_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 “tpmc500api.h”

TPMC500_HANDLE hdl;
TPMC500_STATUS result;

/*
** close file descriptor to device
*/
result = tpmc500Close( hdl );

if (result != TPMC500_OK)
{
     /* handle close error */
}
```

RETURNS

On success TPMC500_OK, or an appropriate error code.

ERROR CODES

All error codes are standard error codes set by the I/O system.

3.2 Device Access Functions

3.2.1 tpmc500Read

NAME

tpmc500Read – Read converted AD value

SYNOPSIS

```
TPMC500_STATUS tpmc500Read
(
      TPMC500_HANDLE          hdl,
      int                     channel,
      int                     gain,
      int                     flags,
      int                     *pAdcVal
);
```

DESCRIPTION

This function starts an AD conversion on the specified channel and returns the converted value.

PARAMETERS

hdl

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

channel

This argument specifies the input channel number. Valid channels for single-ended mode are 1…32, for differential mode 1...16.

gain

This argument specifies the gain for this channel. Valid gains are 1, 2, 5, 10 for *TPMC500-10/-12/-20/-22* and 1, 2, 4, 8 for *TPMC500-11/-13/-21/-23.*

flags

Set of bit flags that control the AD conversion. The following flags could be OR'ed:

Flag	Meaning
TPMC500_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC500_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC500 EEPROM.
TPMC500_FAST	If this flag is set the fast (polled) mode will be used. The driver will not use interrupts, instead it will wait in a busy loop until the settling time (if necessary) and the conversion is finished. Conversions using this mode will be handled faster, but the processor executes a busy loop and other tasks will not be handled during the loops.

pAdcVal

This argument points to an integer variable where the AD value will be returned. The 12-bit value is always moved to the least significant bits. The returned value is in the range from 0…4095 for unipolar input and -2048…2047 for bipolar input.

EXAMPLE

```
#include “tpmc500api.h”

TPMC500_HANDLE   hdl;
TPMC500_STATUS   result;
int              AdcData;
int              channel, gain, flags;

channel   = 32;
gain      = 2;
flags     = TPMC500_CORR | TPMC500_FAST;

result = tpmc500Read(hdl, channel, gain, flags, &AdcData);

if (result != TPMC500_OK)
{
    /* handle error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_ACCESS	The module type has not been configured.
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.
TPMC500_ERR_INVAL	At least one of the parameters is invalid.
TPMC500_ERR_TIMEOUT	ADC conversion timed out.
TPMC500_ERR_RANGE	Invalid channel number.
TPMC500_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.2 tpmc500StartSequencer

NAME

tpmc500StartSequencer – Start sequencer operation

SYNOPSIS

```
TPMC500_STATUS tpmc500StartSequencer
(
        TPMC500_HANDLE       hdl,
        unsigned int         CycleTime,
        unsigned int         NumOfBufferPages,
        unsigned int         NumOfChannels,
        TPMC500_CHAN_CONF    *ChanConf
);
```

DESCRIPTION

This function sets up and starts the sequencer. The setup specifies the channels to be used in sequencer mode and how they will be setup, defining gain, correction and input interface. Additional the sequencer cycle time is defined and depth of the drivers sequencer FIFO will be configured.

PARAMETERS

hdl

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

CycleTime

This argument specifies the repeat frequency of the sequencer in 100 µs steps. Each time the sequencer timer reaches the programmed cycle time a new AD conversion of all active channels is started. Valid values are in the range from 100 microseconds to 6.5535 seconds.

NumOfBufferPages

This argument specifies the number of sample blocks in the ring buffer. A sample block contains the samples of all channels (NumOfChannels) per sequencer cycle.

NumOfChannels

This argument specifies the number of active channels for this job. The maximum number is 32.

ChanConf

This array of channel configuration structures specifies the configuration of the active channels. The channel configuration defines the channel number, the gain and some flags. The ordering of channels in a ring buffer page is the same as defined in this array.

```
typedef struct
{
    UINT32    ChanToUse;
    UINT32    gain;
    UINT32    flags;
} TPMC500_CHAN_CONF, *PTPMC500_CHAN_CONF;
```

ChanToUse

This parameter specifies the input channel number. Valid channels for single-ended mode are 1…32, for differential mode 1...16.

gain

This Parameter specifies the gain for this channel. Valid gains are 1, 2, 5, 10 for *TPMC500-10/-12/-20/-22* and 1, 2, 4, 8 for *TPMC500-11/-13/-21/-23.*

flags

Set of bit flags that control the AD conversion. The following flags could be OR'ed:

Flag	**Meaning**
TPMC500_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC500_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC500 EEPROM.

EXAMPLE

```
#include "tpmc500api.h"

TPMC500_HANDLE     hdl;
TPMC500_STATUS     result;
unsigned int       CycleTime;
unsigned int       NumOfBufferPages;
unsigned int       NumOfChannels;
TPMC500_CHAN_CONF  ChanConf[TPMC500_MAX_CHAN];

CycleTime = 5000;
NumOfBufferPages = 100;
int NumOfChannels = 2;

ChanConf[0].ChanToUse   = 1;
ChanConf[0].gain        = 1;
ChanConf[0].flags       = TPMC500_CORR;

ChanConf[1].ChanToUse   = 20;
ChanConf[1].gain        = 5;
ChanConf[1].flags       = TPMC500_CORR;
…
```

```
// start the sequencer
result = tpmc500StartSequencer(hdl, CycleTime, NumOfBufferPages,
                               NumOfChannels, ChanConf);

if (result != TPMC500_OK)
{
    /* handle error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_ACCESS	The module type has not been configured.
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.
TPMC500_ERR_INVAL	At least one of the parameters is invalid.
TPMC500_ERR_RANGE	Invalid channel number.
TPMC500_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.3 tpmc500GetDataBuffer

NAME

tpmc500GetDataBuffer – Get next data block of sequencer samples

SYNOPSIS

```
TPMC500_STATUS tpmc500GetDataBuffer
(
        TPMC500_HANDLE          hdl,
        int                     **pData,
        unsigned int            *pStatus
);
```

DESCRIPTION

This function returns a pointer to the next available data block in the ring buffer. If no data block is available the functions returns TPMC500_ERR_NODATA. In this case it must be called again until new data is available.

PARAMETERS

hdl

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

pData

This argument is a pointer to an array of integer items that contains the converted data of all configured channels of a sequencer cycle. The number of channels and the channel configuration was setup with the tpmc500StartSequencer function. The first array item [0] belongs to the channel configured by ChanConfig[0], the second array item [1] belongs to the channel configured by ChanConfig[1] and so forth. Please refer to the example application for details.

pStatus

This argument is a pointer to a variable which returns the actual sequencer error status. Keep in mind to check this status before each reading. If status is 0 no error is pending. A set of bits specifies the error condition.

Value	**Description**
TPMC500_BUF_OVERRUN	This bit indicates a ring buffer overrun. The error occurred if there is no space in ring buffer to write the new AD data. In this case the new AD values are dismissed. The sequencer was not stopped.
TPMC500_DATA_OVERFLOW	This indicates an overrun in the sequencer data RAM. The error occurred if the driver is too slow to read the data in time. The sequencer was stopped after this error occurred.
TPMC500_TIMER_ERR	Sequencer timer error (see also TPMC500 hardware manual). The sequencer was stopped after this error occurred.
TPMC500_INST_RAM_ERR	Sequencer instruction RAM error (see also TPMC500 hardware manual). The sequencer was stopped after this error occurred.

EXAMPLE

```
#include “tpmc500api.h”

TPMC500_HANDLE   hdl;
TPMC500_STATUS   result;
unsigned int     seqStatus;
int              *pData;


result = tpmc500GetDataBuffer(hdl, &pData, &seqStatus);

if (result != TPMC500_OK)
{
     if (result == TPMC500_ERR_NODATA)
     {
          /* try again reading data */
     }
     else
     {
          /* handle error */
     }
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.
TPMC500_ERR_NODATA	No new data available in the ring buffer
TPMC500_ERR_NOT_READY	The sequencer is stopped.

3.2.4 tpmc500StopSequencer

NAME

tpmc500StopSequencer – Stop the sequencer

SYNOPSIS

```
TPMC500_STATUS tpmc500StopSequencer
(
      TPMC500_HANDLE          hdl
);
```

DESCRIPTION

This function stops execution of the sequencer mode on the specified device.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc500api.h”

TPMC500_HANDLE   hdl;
TPMC500_STATUS   result;

result = tpmc500StopSequencer(hdl);

if (result != TPMC500_OK)
{
    /* handle error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.

3.2.5 tpmc500SetModelType

NAME

tpmc500SetModelType – Set the module type of the TPMC500

SYNOPSIS

```
TPMC500_STATUS tpmc500SetModelType
(
        TPMC500_HANDLE          hdl,
        int                     ModuleType
);
```

DESCRIPTION

This TPMC500 function configures the model type of the TPMC500.

This function must be called before the first AD conversion can be started.

PARAMETERS

hdl

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

ModuleType

This argument specifies the model type of the TPMC500. The following model types are supported.

Value	**Description**
TPMC500_TYPE_10	TPMC500-10 (Gain 1/2/5/10, +/-10V, Front I/O)
TPMC500_TYPE_11	TPMC500-11 (Gain 1/2/4/8, +/-10V, Front I/O)
TPMC500_TYPE_12	TPMC500-12 (Gain 1/2/5/10, 0-10V, Front I/O)
TPMC500_TYPE_13	TPMC500-13 (Gain 1/2/4/8, 0-10V, Front I/O)
TPMC500_TYPE_20	TPMC500-20 (Gain 1/2/5/10, +/-10V, Back I/O)
TPMC500_TYPE_21	TPMC500-21 (Gain 1/2/4/8, +/-10V, Back I/O)
TPMC500_TYPE_22	TPMC500-22 (Gain 1/2/5/10, 0-10V, Back I/O)
TPMC500_TYPE_23	TPMC500-23 (Gain 1/2/4/8, 0-10V, Back I/O)

EXAMPLE

```
#include “tpmc500api.h”

TPMC500_HANDLE   hdl;
TPMC500_STATUS   result;

result = tpmc500SetModelType(hdl, TPMC500_TYPE_11);

if (result != TPMC500_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.
TPMC500_ERR_RANGE	Invalid channel number.

3.2.6 tpmc500GetModuleInfo

NAME

tpmc500GetModuleInfo – Get module information data

SYNOPSIS

```
TPMC500_STATUS tpmc500GetModuleInfo
(
    TPMC500_HANDLE          hdl,
    TPMC500_INFO_BUFFER     *pModuleInfo
);
```

DESCRIPTION

This function reads module information data such as configured module type, location on the PCI bus and factory programmed correction data.

PARAMETERS

hdl

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

pModuleInfo

This argument specifies a pointer to the module information buffer.

```
typedef struct
{
   UINT32     Variant;
   UINT32     PciBusNo;
   UINT32     PciDevNo;
   UINT32     ADCOffsetCal[4];
   UINT32     ADCGainCal[4];
} TPMC500_INFO_BUFFER, *PTPMC500_INFO_BUFFER;
```

Variant

This parameter returns the configured module variant (e.g. 10 for a TPMC500-10).

PciBusNo, PciDevNo

These parameters specifies the PCI location of this module

ADCOffsetCal[4]

This array returns the factory programmed offset correction value for the different gains. Array index 0 contains the value for gain 1, index 1 contains the value for gain 2 and so forth.

ADCGainCal[4]

This array returns the factory programmed gain correction for the different gains. Array index 0 contains the value for gain 1, index 1 contains the value for gain 2 and so forth.

EXAMPLE

```
#include “tpmc500api.h”

TPMC500_HANDLE        hdl;
TPMC500_STATUS        result;
TPMC500_INFO_BUFFER   ModuleInfo

result = tpmc500GetModuleInfo(hdl, &ModuleInfo);

if (result != TPMC500_OK)
{
    /* handle error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.