The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC550-SW-65

Windows Device Driver

8/4 Channels of Isolated 12-Bit D/A

Version 2.0.x

User Manual

Issue 2.0.0

June 2011

powerBridge Computer

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

www.powerbridge.de
info@powerbridge.de

WS TECHNOLOGIES GmbH
ı Bahnhof 7 25469 Halstenbek, Germany
101 4058 0 Fax: +49 (0) 4101 4058 19
@tews.com www.tews.com

TPMC550-SW-65

Windows Device Driver

8/4 Channels of Isolated 12-Bit D/A

Supported Modules:
TPMC550

Issue	Description	Date
1.0	First Issue	March 28, 2003
1.0.1	General revision, new address TEWS LLC	November 11, 2008
2.0.0	Windows7 support and API functions added	June 14, 2011

Table of Contents

1 Introduction

The TPMC550-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 TPMC550-SW-65 device driver supports the following features:

- writing and converting D/A values to a specified channel
- simultaneous D/A conversion on selected channels
- sequencer facility with configurable software ring buffer
- reading module configuration (voltage range and correction data)

The TPMC550-SW-65 supports the modules listed below:

TPMC550	8/4 Channels of Isolated 12 bit D/A	PMC

To get more information about the features and use of TPMC550 devices it is recommended to read the manuals listed below.

TPMC550 User manual

2 Installation

Following files are located in directory TPMC550-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)
tpmc550.inf	Windows installation script
tpmc550.h	Header file with IOCTL codes and structure definitions
example\tpmc550exa.c	Example application
api\tpmc550api.c	Application Programming Interface source
api\tpmc550api.h	Application Programming Interface header
TPMC550-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 / XP

This section describes how to install the TPMC550 Device Driver on a Windows 2000 / XP operating system.

After installing the TPMC550 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 TPMC550 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 (tpmc550.h and API files) to the desired target directories.

After successful installation the TPMC550 device driver will start immediately and creates devices (TPMC550_1, TPMC550_2 ...) for all recognized TPMC550 modules.

2.1.2 Windows 7

This section describes how to install the TPMC550-SW-65 Device Driver on a Windows 7 (32bit or 64bit) 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 (tpmc550.h and API files) to desired target directory.

After successful installation a device is created for each module found (TPMC550_1, TPMC550_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 – TPMC550 8(4) Channel 12-Bit D/A (TPMC550)***" should appear for each installed device.

3 API Documentation

3.1 General Functions

3.1.1 tpmc550Open

NAME

tpmc550Open – Opens a Device

SYNOPSIS

```
TPMC550_HANDLE tpmc550Open
(
      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 "tpmc550api.h"

TPMC550_HANDLE hdl;

/*
** open file descriptor to device
*/
hdl = tpmc550Open("\\\\.\\TPMC550_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 tpmc550Close

NAME

tpmc550Close – Closes a Device

SYNOPSIS

```
TPMC550_STATUS tpmc550Close
(
      TPMC550_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 "tpmc550api.h"

TPMC550_HANDLE hdl;
TPMC550_STATUS result;

/*
** close file descriptor to device
*/
result = tpmc550Close( hdl );

if (result != TPMC550_OK)
{
     /* handle close error */
}
```

RETURNS

On success TPMC550_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 tpmc550DacWrite

NAME

tpmc550DacWrite – write D/A value to specified channel

SYNOPSIS

```
TPMC550_STATUS tpmc550DacWrite
(
    TPMC550_HANDLE      hdl,
    int                 channel,
    unsigned int        flags,
    short               value
);
```

DESCRIPTION

This function writes a new value to a specific channel and starts D/A conversion immediately in transparent mode

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 DAC channel which shall be updated. Possible values are 1 up to the number of available DAC channels of the specific module.

flags

This argument specifies a set of bit flags that control the D/A conversion:

Value	Description
TPMC550_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC550 EEPROM.

value

This argument specifies the new 12-bit D/A value. Valid data range depends on the voltage range of the specified channel (0…4095 for 0...10V voltage range and -2048…2047 for +/-10V voltage range).

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE  hdl;
TPMC550_STATUS  result;



result = tpmc550DacWrite(hdl, 1, TPMC550_CORR, 1234);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.
TPMC550_ERR_RANGE	Invalid channel number
TPMC550_ERR_TIMEOUT	Timeout during D/A conversion
TPMC550_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.2 tpmc550DacWriteMulti

NAME

tpmc550DacWriteMulti – write D/A value to multiple channels

SYNOPSIS

```
TPMC550_STATUS tpmc550DacWriteMulti
(
    TPMC550_HANDLE      hdl,
    unsigned int        ChannelMask,
    unsigned int        flags,
    short               values[MAX_CHAN]
);
```

DESCRIPTION

This function writes new values to specified channels and starts D/A conversion immediately (transparent mode) or simultaneously (latched mode).

PARAMETERS

hdl

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

ChannelMask

This argument selects DAC channels which shall be updated. A set (1) bit specifies that the corresponding channel shall be updated. Bit 0 corresponds to the first DAC channel, bit 1 corresponds to the second DAC channel and so on.

flags

This argument specifies a set of bit flags that control the D/A conversion:

Value	Description
TPMC550_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC550 EEPROM for all selected channels.
TPMC550_SIMCONV	Start conversion of selected channels in latched mode and update analog outputs simultaneously.

values

This array contains the new 12-bit D/A values. Valid data range depends on the voltage range of the specified channel (0…4095 for 0...10V voltage range and -2048…2047 for +/-10V voltage range).

Array index 0 corresponds to the first DAC channel, array index 1 corresponds to the second DAC channel and so on. Only channels selected for update (*ChannelMask)* will be modified.

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE   hdl;
TPMC550_STATUS   result;
unsigned int     ChannelMask;
unsigned int     flags;
short            values[MAX_CHAN];

// Update channel 1, 4 and 8 simultaneously with corrected D/A values
ChannelMask = (1<<0) | (1<<3) | (1<<7);
flags = TPMC550_CORR | TPMC550_SIMCONV;
value[0] = 111;    // channel 1
value[3] = 444;    // channel 4
value[7] = 888;    // channel 8

result = tpmc550DacWriteMulti(hdl, ChannelMask, flags, values);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.
TPMC550_ERR_RANGE	Invalid channel number
TPMC550_ERR_TIMEOUT	Timeout during D/A conversion
TPMC550_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.3 tpmc550SeqSetup

NAME

tpmc550SeqSetup – Setup sequencer facility

SYNOPSIS

```
TPMC550_STATUS tpmc550SeqSetup
(
    TPMC550_HANDLE      hdl,
    int                 CycleTime,
    int                 NumActiveChannels,
    int                 NumBufTuples,
    int                 ChannelAllocation[MAX_CHAN],
    unsigned int        flags
);
```

DESCRIPTION

This function configures the sequencer facility and allocates memory for the sequencer software ring buffer. The behaviour of the sequencer facility is controlled by a set of bit flags which are described below.

Basically the sequencer will perform a D/A conversion on active channels in a deterministic time period controlled by a cycle timer or the duration of the conversion itself. To be sure that D/A data will be available for the next cycle just in-time, data for the sequencer will be provided by a configurable ring buffer. The ring buffer can be asynchronously filled by the application program.

The sequencer facility provides two operating modes. In loop mode (TPMC550_LOOP) the buffer will be filled completely with new data (e.g. wave form). The contents of the buffer will be output continuously in a loop. In normal mode (TPMC550_LOOP is not set) the application program must provide new data for every cycle. If the buffer is empty then the sequencer will stop and it holds the last output value until new data arrives.

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 sequencer cycle time in steps of 100 µs. This argument is only relevant if the flag TPMC550_TIMERMODE is set.

NumActiveChannels

This argument specifies the number of active channels. Valid range is 1 up to the number of available channels (4 or 8).

NumBufTuples

This argument specifies the size of the sequencer software ring buffer. In this case size is not the number of bytes to allocate but rather the number of tuples (data for all active channels per cycle).

ChannelAllocation

This argument specifies the channel number of active channels and their enumeration inside a tuple. The function tpmc550SeqWrite awaits new data for active channels in this order. The first array element contains the channel number (1..n) of the first active channel. The second array element the channel number of the second active channel and so forth. Unused array elements can be left undefined.

flags

This argument specifies a set of bit flags that control the sequencer operation:

Value	Description
TPMC550_TIMERMODE	If set, the cycle of D/A conversions will be controlled by the sequencer timer in steps of 100 microseconds; otherwise the sequencer will run in continuous mode as fast as possible (based on the conversion time).
TPMC550_LOOP	If this flag is set (loop mode) the ring buffer never becomes empty. Once completely filled the sequencer will continuously get data out of the buffer for the next conversion. If this flag is not set (normal mode) and the buffer becomes empty then the sequencer will stop and it holds the last output value until new data arrives.
TPMC550_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC550 EEPROM for all selected channels.
TPMC550_SIMCONV	Start conversion of active channels in latched mode and update analog outputs simultaneously.

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE   hdl;
TPMC550_STATUS   result;
int              ChannelAllocation[MAX_CHAN];
unsigned int     flags;

// Setup the sequencer with 2 active channels (1 and 4) in timer mode with
// 1 ms cycle time. The sequencer buffer shall store data tuples for
// up to 100 cycles.

ChannelAllocation[0] = 1;
```

```
ChannelAllocation[1] = 4;
flags = TPMC550_TIMERMODE | TPMC550_CORR | TPMC550_SIMCONV;

result = tpmc550SeqSetup(hdl, 10, 2, 100, ChannelAllocation, flags);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.
TPMC550_ERR_RANGE	Invalid channel number or invalid number of channels.
TPMC550_ERR_NOMEM	Unable to allocate memory for the ring buffer.
TPMC550_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

3.2.4 tpmc550SeqStart

NAME

tpmc550SeqStart – start sequencer facility

SYNOPSIS

```
TPMC550_STATUS tpmc550SeqStart
(
      TPMC550_HANDLE              hdl
);
```

DESCRIPTION

This function starts the sequencer facility. Before calling this function the sequencer must be setup with tpmc550SeqSetup und the ring buffer must be filled with tpmc550SeqWrite.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE  hdl;
TPMC550_STATUS  result;


// start the seuencer
result = tpmc550SeqStart(hdl);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.
TPMC550_ERR_NOT_READY	The sequencer facility was not properly configured. Execute the function tpmc550SeqSetup first.
TPMC550_ERR_NODATA	No data is available in the ring buffer to start the sequencer facility. Use the function tpmc550SeqWrite to write at least one data tuple before starting the sequencer.

3.2.5 tpmc550SeqStop

NAME

tpmc550SeqStop – stop the sequencer facility

SYNOPSIS

```
TPMC550_STATUS tpmc550SeqStop
(
      TPMC550_HANDLE              hdl
);
```

DESCRIPTION

This function stops the sequencer facility. All allocated resources (e.g. ring buffer memory) will be freed.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE  hdl;
TPMC550_STATUS  result;


// stop the sequencer
result = tpmc550SeqStop(hdl);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.

3.2.6 tpmc550SeqWrite

NAME

tpmc550SeqWrite – write new sequencer data

SYNOPSIS

```
TPMC550_STATUS tpmc550SeqWrite
(
    TPMC550_HANDLE      hdl,
    int                 size,
    short               *values,
    int                 *WrittenSize
);
```

DESCRIPTION

This function writes new data to the sequencers data buffer. The provided data buffer must always contain new data for all active channels (tuple). The number of tuples per write must be at least one up to “unlimited”. This function will always write as many tuples as possible. If the buffer becomes full the function will return immediately with the error TPMC550_ERR_BUF_FULL. The number of written bytes will be returned in a variable pointed to by WrittenSize.

PARAMETERS

hdl

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

size

This argument specifies the size (in bytes) of the data buffer to write.

values

This argument is a pointer to an array of short variables that contains data for all active channels for at least one sequencer cycle (tuple). Despite of the declaration as simple short pointer this array is treated as a two-dimensional array with variable dimensions. The rows of the array represent the number of tuples and the columns the number of active channels. A declaration of this array will look like this: *data[tuples][channels]*.

WrittenSize

This argument is a pointer to an int variable where the number of written bytes is returned. In case of the error TPMC550_ERR_BUF_FULL this value can be used to adjust the buffer start pointer for subsequent writes.

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE   hdl;
TPMC550_STATUS   result;
int              WrittenSize;
short            ForOneCycle[4];
short            ForHundredCycles[100][4];

// Fill new data into the data buffers
ForHundredCycles[0][0] = 1;      // first cycle, first channel
ForHundredCycles[0][1] = 2;      // first cycle, second channel
// ...
ForHundredCycles[1][0] = 11;     // second cycle, first channel
// ...
ForHundredCycles[99][3] = 1234;  // 100th cycle, last channel

// Write new data for 100 cycles and 4 active channels (100 * 4 values)

result = tpmc550SeqWrite(
               hdl,
               sizeof(ForHundredCycles),
               (short*)ForHundredCycles,
               &WrittenSize
               );

if (result != TPMC550_OK)
{
     /* handle error */
     if (result == TPMC550_ERR_BUF_FULL)
     {
          /* send remaining data later */
     }
}

// Write new data for 1 cycle and 4 active channels (4 values)

result = tpmc550SeqWrite(
               hdl,
               sizeof(ForOneCycle),
               ForOneCycle,
               &WrittenSize
               );
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.
TPMC550_ERR_NOT_READY	The sequencer is not running
TPMC550_ERR_BUF_TOO_SMALL	The buffer does not contain enough data for all active channels.
TPMC550_ERR_NOMEM	The passed data buffer does not fit into the configured sequencer buffer. This error is only relevant in loop mode (TPMC550_LOOP)
TPMC550_ERR_BUF_FULL	The sequencer buffer is full. Not all data was written to the buffer. Use the contents of WrittenSize to adjust the data pointer to write the remaining data tuples.

3.2.7 tpmc550SeqFlush

NAME

tpmc550SeqFlush – flush the sequencer ring buffer

SYNOPSIS

```
TPMC550_STATUS tpmc550SeqFlush
(
      TPMC550_HANDLE              hdl
);
```

DESCRIPTION

This function flushes the ring buffer of the sequencer facility. The analog output of active channels will hold the last converted data until new data is written with the tpmc550SeqWrite function.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE  hdl;
TPMC550_STATUS  result;


// flush the sequencer ring buffer
result = tpmc550SeqFlush(hdl);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.

3.2.8 tpmc550SeqStatus

NAME

tpmc550SeqStatus – get sequencer status and statistic information

SYNOPSIS

```
TPMC550_STATUS tpmc550SeqStatus
(
    TPMC550_HANDLE      hdl,
    int                 *OperatingState,
    int                 *status,
    int                 *CycleCount,
    int                 *UnderflowCount,
    int                 *EmptyCount
);
```

DESCRIPTION

This function reads sequencer status and statistic information from the specified device.

PARAMETERS

hdl

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

OperatingState

This argument is a pointer to an int variable where the current operating state of the sequencer is returned. Possible operating states are:

Value	Description
TPMC550_STOPPED	The sequencer is stopped.
TPMC550_READY	The sequencer facility is configured and ready to start.
TPMC550_RUNNING	The sequencer is running.

status

This argument is a pointer to an int variable where current error/status of the sequencer is returned. After calling this function the error/status code will be set to TPMC550_SEQ_OK. Possible error/status codes are:

Value	Description
TPMC550_SEQ_OK	Sequencer is working fine. No errors detected.
TPMC550_SEQ_UNDERFLOW	The sequencer hardware has detected a data underflow condition. The driver was not able to provide new data within a sequencer timer cycle.
TPMC550_SEQ_NODATA	No data available in the ring buffer for output.

CycleCount

This argument is a pointer to an int variable where the total number of sequencer cycles since sequencer start is returned.

UnderflowCount

This argument is a pointer to an int variable where the total number of sequencer underflows since sequencer start is returned.

EmptyCount

This argument is a pointer to an int variable where the total number of empty buffer cycles since sequencer start is returned.

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE   hdl;
TPMC550_STATUS   result;
int              OperatingState;
int              status;
int              CycleCount;
int              UnderflowCount;
int              EmptyCount;

// Read sequencer status and statistic information

result = tpmc550SeqStatus(hdl, &OperatingState, &status, &CycleCount,
                          &UnderflowCount, &EmptyCount);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.

3.2.9 tpmc550GetModuleInfo

NAME

tpmc550GetModuleInfo – Get module information

SYNOPSIS

```
TPMC550_STATUS tpmc550GetModuleInfo
(
    TPMC550_HANDLE      hdl,
    int                 *NumChan,
    int                 bipolar[MAX_CHAN],
    char                OffsCorr[MAX_CHAN],
    char                GainCorr[MAX_CHAN]
);
```

DESCRIPTION

This function reads module information data from the specified device.

PARAMETERS

hdl

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

NumChan

This argument is a pointer to an int variable where the number of available DAC channels is returned.

bipolar

This argument is a pointer to an int array where the configured voltage range of each DAC channel is returned as boolean value. The array element bipolar[0] contains the range stetting for DAC channel 1, bipolar[1] for DAC channel 2 and so forth. If the corresponding value is TRUE then the voltage range of the channel is configured to +/- 10V output (bipolar); otherwise it is configured to 0…10V output voltage range.

OffsCorr

This argument is a pointer to a char array where the factory programmed offset correction data is returned. OffsCorr[0] contains correction data for DAC channel 1, OffsCorr[1] for DAC channel 2 and so forth.

GainCorr

This argument is a pointer to a char array where the factory programmed gain correction data are returned. GainCorr[0] contains correction data for DAC channel 1, GainCorr[1] for DAC channel 2 and so forth.

EXAMPLE

```
#include "tpmc550api.h"

TPMC550_HANDLE   hdl;
TPMC550_STATUS   result;
int              NumChan;
int              bipolar[MAX_CHAN];
char             OffsCorr[MAX_CHAN];
char             GainCorr[MAX_CHAN];

// Get module information data

result = tpmc550GetModuleInfo(hdl, &NumChan, bipolar, OffsCorr, GainCorr);

if (result != TPMC550_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC550_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC550_ERR_INVALID_HANDLE	The specified TPMC550_HANDLE is invalid.