The Embedded I/O Company

TPMC551-SW-65

Windows 2000 / XP Device Driver

8/4 Channel of Isolated 16-bit D/A

Version 1.1.x

User Manual

Issue 1.1.1

January 2010

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
) 4101 4058 0 Fax: +49 (0) 4101 4058 19
fo@tews.com www.tews.com

TPMC551-SW-65

Windows 2000 / XP Device Driver

8/4 Channel of Isolated 16-bit D/A

Supported Modules:
TPMC551

Issue	Description	Date
1.0.0	First Issue	August 4, 2004
1.1.0	General revision, new address of TEWS LLC	June 23, 2009
1.1.1	File list changed	January 26, 2010

Table of Contents

1 Introduction

The TPMC551-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TPMC551 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 TPMC551-SW-65 device driver supports the following features:

- writing and converting DA values to a specified channel
- starting simultaneous conversion of latched values on all channels
- reading the hardware configuration (voltage range and correction data) of a specified channel
- configuring and starting the sequencer in different modes to output user supplied buffers
- reading sequencer state (overruns and activity)
- configuring the module type
- correction of output values

The TPMC551-SW-65 device driver supports the modules listed below:

TPMC551-10	8 Channel of Isolated 16-bit D/A (front panel I/O)	(PMC)
TPMC551-11	4 Channel of Isolated 16-bit D/A (front panel I/O)	(PMC)
TPMC551-20	8 Channel of Isolated 16-bit D/A (P14 I/O)	(PMC)
TPMC551-21	8 Channel of Isolated 16-bit D/A (P14 I/O)	(PMC)

To get more information about the features and use of TPMC551 devices it is recommended to read the manuals listed below.

TPMC551 User manual

TPMC551 Engineering Manual

2 Installation

Following files are located on the distribution media:

Directory path 'TPMC551-SW-65':

tpmc551.sys	Windows driver binary
tpmc551.h	TPMC551 include file for driver and application
tpmc551.inf	Windows installation script
example/tpmc551exa.c	Microsoft Visual C example application
EmbeddedIoDeviceClass.dll	Windows WDM device class library
TPMC551-SW-65-1.1.1.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TPMC551 Device Driver on a Windows 2000 / XP operating system.

After installing the TPMC551 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. In Drive A, insert the TPMC551 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.

After successful installation the TPMC551 device driver will start immediately and creates devices (TPMC551_1, TPMC551_2 ...) for all recognized TPMC551 modules.

2.1.2 Confirming Windows 2000 / XP Installation

To confirm that the driver has been properly loaded in Windows 2000 / XP, perform the following steps:

1. From Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***".
2. Click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
3. Click the "**+**" in front of "***Embeded I/O***".
   The driver "***TEWS TECHNOLOGIES - TPMC551 8(4) Channel 16 bit DAC***" should appear.

3 TPMC551 Device Driver Programming

The TPMC551-SW-65 Windows WDM device driver is a kernel mode device driver.

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

3.1 TPMC551 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TPMC551device driver. Only the required parameters are described in detail.

3.1.1 Opening a TPMC551 Device

Before you can perform any I/O the *TPM551* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TPMC551* device.

```
HANDLE CreateFile(
        LPCTSTR   lpFileName,
        DWORD     dwDesiredAccess,
        DWORD     dwShareMode,
        LPSECURITY_ATTRIBUTES lpSecurityAttributes,
        DWORD     dwCreationDistribution,
        DWORD     dwFlagsAndAttributes,
        HANDLE    hTemplateFile
);
```

Parameters

LPCTSTR lpFileName

This parameter points to a null-terminated string, which specifies the name of the TPMC551 to open. The *lpFileName* string should be of the form **\\.\TPMC551_*x*** to open the device *x.* The ending x is a one-based number. The first device found by the driver is \\.\TPMC551_1, the second \\.\TPMC551_2 and so on.

DWORD dwDesiredAccess

This parameter specifies the type of access to the TPMC551.
For the TPMC551 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

DWORD dwShareMode

Set of bit flags that specify how the object can be shared. Set to 0.

LPSECURITY_ATTRIBUTES
lpSecurityAttributes

This argument is a pointer to a security structure. Set to NULL for TPMC551 devices.

DWORD *dwCreationDistribution*

Specifies the action to take on existing files, and which action to take when files do not exist. TPMC551 devices must be always opened **OPEN_EXISTING**.

DWORD *dwFlagsAndAttributes*

Specifies the file attributes and flags for the file. This value must be set to 0 (no overlapped I/O).

HANDLE *hTemplateFile*

This value must be NULL for TPMC551 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TPMC551 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TPMC551_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                // no security attrs
     OPEN_EXISTING,       // TPMC551 device always open existing
     0,                   // no overlapped I/O
     NULL
);

if (hDevice == INVALID_HANDLE_VALUE) {
     ErrorHandler( "Could not open device" ); // process error
}
```

See Also

CloseHandle(), Win32 documentation CreateFile()

3.1.2 Closing a TPMC551 Device

The **CloseHandle** function closes an open TPMC551 handle.

```
BOOL CloseHandle(
        HANDLE hDevice;
);
```

Parameters

HANDLE *hDevice*

Identifies an open TPMC551 handle.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call ***GetLastError*.**

Example

```
HANDLE  hDevice;

if( !CloseHandle( hDevice ) ) {
     ErrorHandler( "Could not close device" ); // process error
}
```

See Also

CreateFile (), Win32 documentation CloseHandle ()

3.1.3 TPMC551 Device I/O Control Functions

The **DeviceIoControl** function sends a control code directly to a specified device driver, causing the corresponding device to perform the specified operation.

```
BOOL DeviceIoControl(
        HANDLE              hDevice,            // handle to device of interest
        DWORD               dwIoControlCode,    // control code of operation to perform
        LPVOID              lpInBuffer,         // pointer to buffer to supply input data
        DWORD               nInBufferSize,      // size of input buffer
        LPVOID              lpOutBuffer,        // pointer to buffer to receive output data
        DWORD               nOutBufferSize,     // size of output buffer
        LPDWORD             lpBytesReturned,    // pointer to variable to receive output byte count
        LPOVERLAPPED        lpOverlapped        // pointer to overlapped structure for
                                                // asynchronous operation
);
```

Parameters

hDevice

Handle to the TPMC551 that is to perform the operation.

dwIoControlCode

Specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in *tpmc551.h* :

Value	**Meaning**
IOCTL_TP551_WRITE	Write and convert DA value
IOCTL_TP551_SIMLOAD	Execute a simultaneous load on all channels
IOCTL_TP551_READ_CONF	Read the channel configuration
IOCTL_TP551_SETUP_SEQ	Setup and start sequencer
IOCTL_TP551_STOP_SEQ	Stop sequencer
IOCTL_TP551_DATA_SEQ	Transfer data for a channel in sequencer mode
IOCTL_TP551_STAT_SEQ	Read status of sequencer mode
IOCTL_TP551_CONF_MOD_TYPE	Configure which model type is mounted

See below for more detailed information on each control code.

To use these TPMC551 specific control codes the header file tpmc551.h must be included in the application

lpInBuffer

Pointer to a buffer that contains the data required to perform the operation.

nInBufferSize

Specifies the size of the buffer pointed to by *lpInBuffer*.

lpOutBuffer

Pointer to a buffer that receives the operation's output data.

nOutBufferSize

Specifies the size of the buffer in bytes pointed to by *lpOutBuffer*.

lpBytesReturned

Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by *lpOutBuffer*. A valid pointer is required.

lpOverlapped

Pointer to an *overlapped* structure. This parameter must be NULL because the TPMC551 device driver does not use overlapped I/O.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call ***GetLastError***.

See Also

Win32 documentation DeviceIoControl()

3.1.3.1 IOCTL_TP551_WRITE

This TPMC551 control function writes an application supplied long value (16 bit sign extended) to the specified DAC and starts a DA conversion. The *lpInBuffer* parameter passes a pointer to a channel write structure (*TP551_CHAN_WRITE*) to the driver which contains parameters required to perform the operation. The *lpOutBuffer* parameter will not be used. This function can not be called if the sequencer is started.

```
typedef struct {
        ULONG       ChanToUse;        // channel number to use 1...8 (4)
        ULONG       flags;            // TP551_CORR | TP551_LATCH
        long        value;            // new output value
} TP551_CHAN_WRITE, *PTP551_CHAN_WRITE;
```

ChanToUse

Specifies the channel number at which to start the DA conversion. Valid channels are 1…8 for TPMC551-10/-20 and 1...4 for TPMC551-11/-21.

flags

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

Flag	Meaning
TP551_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC551 EEPROM.
TP551_LATCH	Load value into DAC but do not start the conversion. (The conversion will be started with the *IOCTL_TP551_SIMLOAD* command.

value

Specifies the value which will be loaded to the specified channel. This is a sign extended value. Allowed values are 0...65535 if the channel is configured for 0V...10V voltage range and -32768...32767 if it is configured for -10V…+10V voltage range.

Example

```
#include "TPMC551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
TP551_CHAN_WRITE ChanWrite;

…
```

```
…

//
// Start conversion at channel 1, write 0x4000 and correct the
// output with the factory calibration data
//
ChanWrite.ChanToUse      = 1;
ChanWrite.value          = 0x4000;
ChanWrite.flags          = TP551_CORR;

success = DeviceIoControl (
     hDevice,                    // TPMC551 handle
     IOCTL_TP551_WRITE,          // write DA value
     &ChanWrite,                 // conversion parameters
     sizeof( ChanWrite ),        // size of ChanWrite structure
     NULL,                       // no output buffer
     0,                          // size of output buffer
     &NumWritten,                // unused but required
     NULL                        // unused but required
);

if( success ) {
     printf( "Success\n" );
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.2 IOCTL_TP551_SIMLOAD

This TPMC551 control function starts a conversion on all channels of the TPMC551. The last loaded values will be used. The *lpInBuffer* parameter and *lpOutBuffer* will not be used. This function can not be called if the sequencer is started.

Example

```
#include "tpmc551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;

//
// Load 0x4000 to channel 1, …, but do not start the conversion
//
ChanWrite.ChanToUse      = 1;
ChanWrite.value          = 0x4000;
ChanWrite.flags          = TP551_LATCH;

success = DeviceIoControl (
     hDevice,                  // TPMC551 handle
     IOCTL_TP551_WRITE,        // write DA value
     &ChanWrite,               // conversion parameters
     sizeof( ChanWrite ),      // size of ChanWrite structure
     NULL,                     // no output buffer
     0,                        // size of output buffer
     &NumWritten,              // unused but required
     NULL                      // unused but required
);

…
```

```
…

//
// Now start the conversion for all channels
//
success = DeviceIoControl (
     hDevice,                    // TPMC551 handle
     IOCTL_TP551_SIMLOAD,        // Simultaneous Conversion
     NULL,                       // no input buffer
     0,                          // size of input buffer
     NULL,                       // no output buffer
     0,                          // size of output buffer
     &NumWritten,                // unused but required
     NULL                        // unused but required
);

if( success ) {
     printf( "Success\n" );
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.3 IOCTL_TP551_READ_CONF

This TPMC551 control function reads the configuration and the correction data of the selected channel. The *lpInBuffer* and *lpOutBuffer* parameter passes a pointer to a channel configuration structure (*TP551_CHAN_CONF*) to the driver which contains parameters required to perform the operation and the results will be filled in.

```
typedef struct {
        ULONG       ChanToUse;//  channel number to use 1...8 (4)
        ULONG       config;     //  TP551_BIPOL
        short       CalOffset;  //  Offset Calibration Value
        short       CalGain;    //  Gain Calibration Value
} TP551_CHAN_CONF, *PTP551_CHAN_CONF;
```

ChanToUse

Specifies the channel which configuration will be returned. Valid channels for are 1...8 for TPMC551-10/-20 and 1...4 for TPMC551-11/-21.

config

The flag *TP551_BIPOL* will be set if the channel is configured for –10V...10V voltage range. If the flag in not set, the module is configured for 0V...10V voltage range.

CalOffset

These parameters return the factory stored calibration data for offset correction of the specified channel and selected voltage range. These values will be used if data correction is selected.

CarGain

These parameters return the factory stored calibration data for gain correction of the specified channel and selected voltage range. These values will be used if data correction is selected.

Example

```
#include "tpmc551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
TP551_CHAN_CONF ChanConf;

…
```

```
…

//
// Get configuration of channel 1
//
ChanConf.ChanToUse      = 1;

success = DeviceIoControl (
     hDevice,                // TPMC551 handle
     IOCTL_TP551_READ_CONF,  // read channel configuration
     &ChanConf,              // configuration structure
     sizeof( ChanConf ),     // size of ChanConf structure
     &ChanConf,              // configuration values
     sizeof( ChanConf ),     // size of ChanConf structure
     &NumWritten,            // unused but required
     NULL                    // unused but required
);

if( success ) {
     printf( "Success\n" );
     printf( "    Voltage Range: %s..+10V\n",
          (ChanConf.config & TP551_BIPOL) ?
          "-10V" : "0V");
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.4 IOCTL_TP551_SETUP_SEQ

This TPMC551 control function sets up and starts the sequencer mode. The *lpInBuffer* parameter passes a pointer to a sequencer setup structure (*TP551_SEQ_SETUP*) to the driver which contains parameters required to perform the operation. The *lpOutBuffer* parameter will not be used. This function can not be called if the sequencer is already started.

```
typedef struct {
        ULONG       mode;               // TP551_TIMERMODE
        ULONG       flags[8];           // for the channels [TP551_CORR]
        USHORT      cycle;              // cycletime in steps of 100us
        ULONG       ChanToEnable;       // bitmasks which channels are to enable
} TP551_SEQ_SETUP, *PTP551_SEQ_SETUP;
```

mode

This is bit field where the sequencer mode is selected. The following flags can be OR'ed.

Flag	**Meaning**
TP551_TIMERMODE	Must be set if the cycle time should be used as time base to convert. If this flag is not set, the sequencer will try to convert a new value immediately after completing the last conversion.
TP551_1SHOTMODE	This flag specifies that the provided buffer should be converted and afterwards the sequencer will not convert new values. If the flag is not set, the sequencer will renew the DAC values with every cycle, if the end of the provided buffer is reached, the buffer will be used again, or a new buffer will be used (if present). Note that the module will stay in sequencer mode also if all buffers are finished.

flags[]

This array specifies flags for the channels, flags[0] for channel 1, flags[1] for channel 2 and so on. The only flag, that can be set is *TP551_CORR*, which is set to perform an offset and gain correction with factory calibration data stored in the TPMC551 EEPROM.

cycle

Specifies the length of a sequencer cycle. The time is specified in steps of 100µs.

ChanToEnable

This parameter specifies which channel(s) shall be used in sequencer mode. This value is a bit field, Bit 0 must be set to enable Channel1, Bit 1 for Channel 2 and so on. Data buffers must be loaded for enabled channels before the sequencer is started.

Example

```
#include “tpmc551.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
TP551_SEQ_SETUP SeqSetup;

//
// Load data into buffers for used channels
//

…

//
// Start the sequencer with a cycle time of one second and
// repeat the buffers
// Channel 1 and 2 shall convert raw data
// Channel 6 shall use corrected data
//
SeqSetup.mode            = TP551_TIMERMODE;
SeqSetup.cycle           = 10000;
SeqSetup.ChanToEnable    = (1 << 0) | (1 << 1) | (1 << 5);
SeqSetup.flags[0]        = 0;
SeqSetup.flags[1]        = 0;
SeqSetup.flags[5]        = TP551_CORR;

success = DeviceIoControl (
     hDevice,                    // TPMC551 handle
     IOCTL_TP551_SETUP_SEQ,      // setup and start sequencer
     &SeqSetup,                  // conversion parameters
     sizeof( SeqSetup ),         // size of ChanWrite structure
     NULL,                       // no output buffer
     0,                          // size of output buffer
     &NumWritten,                // unused but required
     NULL                        // unused but required
);

…
```

```
…

if( success ) {
     printf( "Success\n" );
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.5 IOCTL_TP551_STOP_SEQ

This TPMC551 control function stops the sequencer immediately. The *lpInBuffer* parameter and *lpOutBuffer* will not be used. This function can not be called if the sequencer is started.

Example

```
#include "tpmc551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;

success = DeviceIoControl (
     hDevice,                    // TPMC551 handle
     IOCTL_TP551_STOP_SEQ,       // Stop the sequencer
     NULL,                       // no input buffer
     0,                          // size of input buffer
     NULL,                       // no output buffer
     0,                          // size of output buffer
     &NumWritten,                // unused but required
     NULL                        // unused but required
);

if( success ) {
     printf( "Success\n" );
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.6 IOCTL_TP551_DATA_SEQ

This TPMC551 control function loads the content of a specified data buffer into the driver. The *lpInBuffer* parameter passes a pointer to a buffer starting with the sequencer data structure (*TP551_SEQ_DATA*) to the driver which contains parameters and data required to perform the operation. The *lpOutBuffer* parameter will not be used. This function can not be called if the sequencer is started. Note that the *TP551_SEQ_DATA* structure is a part of the buffer and that the size of the buffer depends on the number data words.

```
typedef struct {
      ULONG      ChanToUse;      // channel number to use 1..8 (4)
      ULONG      size;           // length of the following buffer
      long       buffer[1];      // first element of buffer
} TP551_SEQ_DATA, *PTP551_SEQ_DATA;
```

ChanToUse

Specifies the channel that should use the data. Valid channels are 1..8 for TPMC551-10/-20 and 1..4 for TPMC551-11/-21.

size

Size of the following buffer in data words.

buffer[]

First value of the user allocated buffer.

Example

```
#include "tpmc551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
PTP551_SEQ_DATA pSeqData;
ULONG size;

//
// Load data buffer with 100 data words for channel 1
//

// ***** Step 1: allocate buffer *****
size = 100 * sizeof(long);   // buffer for data values
size -= sizeof(long);        // first data value is contained
                             // in the parameter structure
size += sizeof(TP551_SEQ_DATA);
                             // Add the size of the parameter
                             // structure

…
```

```
…

pSeqData = malloc(size);

// ***** Step 2: fill the structure *****
pSeqData->ChanToUse     = 1;
pSeqData->size          = 100;
memcpy(pSeqData->buffer, USER_DAC_DATA, 100 * sizeof(long));

success = DeviceIoControl (
     hDevice,                  // TPMC551 handle
     IOCTL_TP551_DAT_SEQ,      // load sequencer data buffer
     pSeqData,                 // sequencer data load buffer
     size,                     // size of data load buffer
     NULL,                     // no output buffer
     0,                        // size of output buffer
     &NumWritten,              // unused but required
     NULL                      // unused but required
);

if( success ) {
     printf( "Success\n" );
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}

// ***** Step 3: return buffer *****
free(pSeqData);
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.7 IOCTL_TP551_STAT_SEQ

This TPMC551 control function returns the current state of the sequencer. The *lpInBuffer* and *lpOutBuffer* parameter passes a pointer to a channel configuration structure (*TP551_SEQ_STATUS*) to the driver where the results will be filled in.

```
typedef struct {
      ULONG      count;// counter of overrun errors
      ULONG      seqActive;    // 0 - sequencer of, else busy
} TP551_SEQ_STATUS, *PTP551_SEQ_STATUS;
```

count

Count returns the number of hardware detected overrun errors occurred while the sequencer is active. (See TPMC551 Hardware User Manual)

seqActive

This value is zero if the sequencer is not active, it is set if it is active.

Example

```
#include "tpmc551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
TP551_SEQ_STATUS SeqStat;

//
// Get sequencer status
//
ChanConf.ChanToUse      = 1;

success = DeviceIoControl (
     hDevice,                      // TPMC551 handle
     IOCTL_TP551_STAT_SEQ,         // read sequencer status
     &SeqStat,                     // configuration structure
     sizeof( SeqStat ),            // size of ChanConf structure
     &SeqStat,                     // configuration values
     sizeof( SeqStat ),            // size of ChanConf structure
     &NumWritten,                  // unused but required
     NULL                          // unused but required
);

…
```

```
…

if( success ) {
     printf( "Success\n" );
     printf( "    Sequencer: %s\n" ,
          SeqStat.seqActive ? "active" : "passive");
     printf( "    Overruns: %d\n" , SeqStat.count);
}
else {    // process error
     ErrorHandler ( "Device I/O control error" );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual

3.1.3.8 IOCTL_TP551_CONF_MOD_TYPE

This TPMC551 control function specifies the modeltype of the TPMC551. The *lpInBuffer* parameter passes a pointer to an unsigned long value to the driver which contains parameters required to perform the operation. The *lpOutBuffer* parameter will not be used. This function can not be called if the sequencer is started. The unsigned long value specifies the model type, the following values are valid:

value	description
TP551_TYPE_10	TPMC551-10 (8Channel, Front I/O)
TP551_TYPE_11	TPMC551-11 (4Channel, Front I/O)
TP551_TYPE_20	TPMC551-20 (8Channel, Back I/O)
TP551_TYPE_21	TPMC551-21 (4Channel, Back I/O)

This function must be called before any other I/O control function is called. This function must be used to tell the driver what kind of TPMC551 is used.

Example

```
#include "TPMC551.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumWritten;
ULONG modelType;

//
// Tell the driver we are using a TPMC551-10
//
modelType = TP551_TYPE_10;

success = DeviceIoControl (
     hDevice,                       // TPMC551 handle
     IOCTL_TP551_CONF_MOD_TYPE,     // write DA value
     &modelType,                    // conversion parameters
     sizeof(modelType),             // size of ChanWrite structure
     NULL,                          // no output buffer
     0,                             // size of output buffer
     &NumWritten,                   // unused but required
     NULL                           // unused but required
);

…
```

```
…

if( success ) {
     printf( “Success\n” );
}
else {    // process error
     ErrorHandler ( "Device I/O control error” );
}
```

See Also

Win32 documentation DeviceIoControl(), TPMC551 Hardware User Manual