The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC851-SW-65

Windows 2000/XP Device Driver

Multifunction I/O (16 bit ADC/DAC, TTL I/O, Counter)

Version 1.0.x

User Manual

Issue 1.0.2

August 2008

TEWS TECHNOLOGIES GmbH		**TEWS TECHNOLOGIES LLC**	
Am Bahnhof 7	Phone: +49-(0)4101-4058-0	9190 Double Diamond Parkway,	Phone: +1 (775) 850 5830
25469 Halstenbek, Germany	Fax: +49-(0)4101-4058-19	Suite 127, Reno, NV 89521, USA	Fax: +1 (775) 201 0347
www.tews.com	e-mail: info@tews.com	www.tews.com	e-mail: usasales@tews.com

TPMC851-SW-65

Windows 2000/XP Device Driver

Multifunction I/O
(16 bit ADC/DAC, TTL I/O, Counter)

Issue	Description	Date
1.0.0	First Issue	January 24, 2005
1.0.1	New Address TEWS LLC, several errors corrected	October 12, 2006
1.0.2	Filelist modified (subdirectory)	August 5, 2008

Table of Contents

1 Introduction

The TPMC851-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TPMC851 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 TPMC851 device driver supports the following features:

- Reading an ADC input value from a specified channel
- Configuring and using the ADC input sequencer
- Setting a DAC output value to a specified channel
- Configuring and using the DAC output sequencer
- Reading from digital I/O input register
- Writing to digital I/O output register
- Waiting for digital I/O input event (high, low or any transition on input line)
- Configuring digital I/O line direction
- Reading counter value
- Reset counter value
- Setting counter preload and match value
- Configuring counter mode
- Wait for counter match and control event

<u>The TPMC851-SW-65 device driver supports the modules listed below:</u>

TPMC851	Multifunction I/O (16 bit ADC/DAC, TTL I/O, Counter)	PMC

To get more information about the features and usage of TPMC851 devices it is recommended to read the manuals listed below.

TPMC851 User Manual

TPMC851 Engineering Manual

2 Installation

Following files are located in directory TPMC851-SW-65 on the distribution media:

tpmc851.sys	Windows driver binary
tpmc851.h	Header-file with IOCTL code definitions
tpmc851.inf	Windows installation script
TPMC851-SW-65-1.0.2.pdf	This document
Example\tpmc851exa.c	Microsoft Visual C example application
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 TPMC851 Device Driver on a Windows 2000 / XP operating system.

After installing the TPMC851 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 TPMC851 driver disk; select "***Disk Drive***" in the dialog box.
   Click "***Next***" button to continue.
4. Now the driver wizard should find a suitable device driver on the diskette.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Now copy all needed files (e.g. tpmc851.h) to the desired target directories.

After successful installation the TPMC851 device driver will start immediately and creates devices (TPMC851_1, TPMC851_2 ...) for all recognized TPMC851 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 "***Other Devices***".
   The driver "***TEWS TECHNOLOGIES TPMC851 (16bit ADC/DAC, TTL I/O, Counter)***" should appear.

3 Driver Configuration

3.1 Event Queue Configuration

After installation of the TPMC851 Device Driver the size of the internal event queues is set to its default value. By default, 10 events can be simultaneously waited for.

If the default value is not suitable, the configuration can be changed by modifying the Windows Registry, for instance with `regedit`.

To change the maximum amount of events to wait for, the following value must be modified:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TPMC51\MaxQueueEntries

4 TPMC851 Device Driver Programming

The TPMC851-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.)

4.1 TPMC851 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TPMC851 device driver. Only the required parameters are described in detail.

4.1.1 Opening a TPMC851 Device

Before you can perform any I/O the *TPMC851* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TPMC851* 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 TPMC851 to open. The *lpFileName* string should be of the form **\\.\TPMC851_*x*** to open the device *x.* The ending x is a one-based number. The first device found by the driver is \\.\TPMC851_1, the second \\.\TPMC851_2 and so on.

DWORD dwDesiredAccess

This parameter specifies the type of access to the TPMC851.
For the TPMC851 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 TPMC851 devices.

DWORD dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TPMC851 devices must be always opened **OPEN_EXISTING**.

DWORD dwFlagsAndAttributes

Specifies the file attributes and flags for the file. This value must be set to FILE_FLAG_OVERLAPPED for TPMC851 devices.

HANDLE hTemplateFile

This value must be NULL for TPMC851 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TPMC851 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TPMC851_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                    // no security attrs
     OPEN_EXISTING,           // TPMC851 device always open existing
     FILE_FLAG_OVERLAPPED,    // overlapped I/O
     NULL
);

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

See Also

CloseHandle(), Win32 documentation CreateFile()

4.1.2 Closing a TPMC851 Device

The **CloseHandle** function closes an open TPMC851 handle.

BOOL CloseHandle(
    HANDLE *hDevice;*
);

Parameters

HANDLE *hDevice*

Identifies an open TPMC851 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 ()

4.1.3 TPMC851 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 TPMC851 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 *tpmc851.h* :

Value	**Meaning**
IOCTL_TP851_ADC_READ	Read value from ADC channel
IOCTL_TP851_ADC_SEQCONFIG	Configure ADC sequencer channel
IOCTL_TP851_ADC_SEQSTART	Start ADC sequencer
IOCTL_TP851_ADC_SEQSTOP	Stop ADC sequencer
IOCTL_TP851_DAC_WRITE	Write value to DAC channel
IOCTL_TP851_DAC_SEQCONFIG	Configure DAC sequencer channel
IOCTL_TP851_DAC_SEQSTART	Start DAC sequencer
IOCTL_TP851_DAC_SEQSTOP	Stop DAC sequencer
IOCTL_TP851_IO_READ	Read from digital I/O
IOCTL_TP851_IO_WRITE	Write to digital I/O
IOCTL_TP851_IO_EVENTWAIT	Wait for I/O event
IOCTL_TP851_IO_CONFIG	Configure digital I/O
IOCTL_TP851_IO_DEBCONFIG	Configure digital I/O (input) debouncer
IOCTL_TP851_CNT_READ	Read value from counter/timer
IOCTL_TP851_CNT_MATCHWAIT	Wait for counter match event
IOCTL_TP851_CNT_CTRLWAIT	Wait for counter control event
IOCTL_TP851_CNT_CONFIG	Configure counter
IOCTL_TP851_CNT_RESET	Reset counter
IOCTL_TP851_CNT_SETPRELD	Set counter preload value
IOCTL_TP851_CNT_SETMATCH	Set counter match value

See below for more detailed information on each control code.

To use these TPMC851 specific control codes the header file tpmc851.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. Refer to the Ioctl specific manual section how this parameter must be set.

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()

4.1.3.1 IOCTL_TP851_ADC_READ

This TPMC851 control function starts an ADC conversion with specified parameters, waits for completion and returns the value. A pointer to the callers buffer (*TP851_ADC_READ_BUF*) must be passed to the driver by *lpOutBuffer* parameter. *lpInBuffer* is not used and should be set to NULL.

The ADC sequencer must be stopped to execute this function.

```
typedef struct
{
        int                     channel;
        int                     gain;
        unsigned long           flags;
        short                   adcValue;
} TP851_ADC_READ_BUF, *PTP851_ADC_READ_BUF;
```

channel

Specifies the ADC channel number. Valid values are 1..16 for differential input and 1..32 for single-ended input.

gain

Specifies the input gain. Valid gain values are 1, 2, 4, and 8.

flags

Is an ored value of the following flags:

flag	description
TP851_F_CORR	If set the function will return a corrected value of the input data in *adcValue*. Factory set and module dependent correction data is used for correction. If not set, the raw value read from the module will be returned in *adcValue*.
TP851_F_IMMREAD	If set the driver will start the conversion without waiting for settling time. This should only be used if the previous conversion has used the same interface parameters (channel, gain, differential/single-ended). If not set the driver will use the automatic mode, which sets interface configuration, waits settling time and then starts the conversion.
TP851_F_DIFF	If set the input channel will be a differential input. If not set the input channel will be a single-ended input.

adcValue

This value will return the read ADC value.

Example

```
#include “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_ADC_READ_BUF      adcReadBuf;

/*
** Read a corrected value from differential channel 2, use a gain of 4
*/
adcReadBuf.channel = 2;
adcReadBuf.gain    = 4;
adcReadBuf.flags   = TP851_F_CORR | TP851_F_DIFF;

printf("Read from ADC ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_ADC_READ,           // control code
     NULL,
     0,
     &adcReadBuf,                    // pointer to buffer
     sizeof(TP851_ADC_READ_BUF),     // size of buffer
     &NumBytes,
     NULL                            // no overlapped I/O
);
if (success)
{
     printf("OK\n    ADC-value: %d\n", adcReadBuf.adcValue);
} else {
     /* handle error */
}
```

Error Codes

ERROR_BUSY	The ADC sequencer is currently running.
ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_INVALID_PARAMETER	Specified gain level is invalid.
ERROR_ACCESS_DENIED	Invalid channel number specified.
ERROR_TIMEOUT	The ADC conversion timed out.

All other returned error codes are system error conditions.

4.1.3.2 IOCTL_TP851_ADC_SEQCONFIG

This TPMC851 control function configures and enables or simply disables an ADC channel for sequencer usage. A pointer to the callers buffer (*TP851_ADC_SEQCONFIG_BUF*) must be passed to the driver by the *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

The ADC sequencer must be stopped to execute this function.

```
typedef struct
{
    int                 channel;
    int                 enable;
    int                 gain;
    unsigned long       flags;
} TP851_ADC_SEQCONFIG_BUF, *PTP851_ADC_SEQCONFIG_BUF;
```

channel

Specifies the ADC channel number to configure. Valid values are 1..16 for differential input and 1..32 for single-ended input.

enable

Specifies if the channel shall be used in sequencer mode or not. (0 disables the channel any other value will enable the channel)

gain

Specifies the input gain. Valid gain values are 1, 2, 4, and 8.

flags

Is an ORed value of the following flags:

flag	**description**
TP851_F_CORR	If set the sequencer will return a corrected value for the specified channel. Factory set and module dependent correction data is used for correction. If not set, the raw value read from the module will be returned.
TP851_F_DIFF	If set the input channel will be a differential input. If not set the input channel will be a single-ended input.

Example

```
#include “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_ADC_SEQCONFIG_BUF adcSeqConfBuf;

/*
** Configure single-ended channel 3, using a gain of 4 and returning
*/
adcSeqConfBuf.channel   = 3;
adcSeqConfBuf.enable    = TRUE;
adcSeqConfBuf.gain      = 4;
adcSeqConfBuf.flags     = TP851_F_CORR;

printf("Configure channel for Sequencer ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_ADC_SEQCONFIG,      // control code
     &adcSeqConfBuf,                 // pointer to buffer
     sizeof(TP851_ADC_SEQCONFIG_BUF),// size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                            // no overlapped I/O
);
if (success)
{
     printf("OK\n");
} else {
     /* handle error */
}
```

Error Codes

ERROR_BUSY	The ADC sequencer is currently running.
ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_ACCESS_DENIED	Invalid channel number specified.
ERROR_INVALID_PARAMETER	Specified gain level or flags invalid.

All other returned error codes are system error conditions.

4.1.3.3 IOCTL_TP851_ADC_SEQSTART

This TPMC851 control function configures the ADC sequencer's cycle time and starts the ADC sequencer. A pointer to the callers buffer (*TP851_ADC_SEQSTART_BUF*) must be passed to the driver by *lpOutBuffer* parameter*. lpInBuffer* is not used and should be set to NULL. This function requires overlapped I/O.

```
typedef struct
{
        unsigned short          cycTime;
        unsigned long           flags;
        long                    putIdx;
        long                    getIdx;
        long                    bufSize;
        long                    seqState;
        short                   buffer[1];
} TP851_ADC_SEQSTART_BUF, *PTP851_ADC_SEQSTART_BUF;
```

cycTime

Specifies the ADC sequencer cycle time. The sequencer time is specified in 100µs steps. With a value of 0, the "Sequencer Continuous Mode" is selected.

flags

Is an ORed value of the following flags:

flag	description
TP851_F_EXTTRIGSRC	If set the ADC sequencer is trigger with digital I/O line 0. If not set, the ADC sequencer uses the ADC cycle counter.
TP851_F_EXTTRIGOUT	If set the ADC trigger is used as output on digital I/O line 0.

TP851_F_EXTTRIGSRC* and *TP851_F_EXTTRIGOUT cannot be used at the same time.

putIdx

Specifies the index into *buffer* where the next data will be stored to. This index is handled by the driver and should only be read by the application to check if data is available. The driver initializes this index to 0 when sequencer starts.

getIdx

Specifies the index into *buffer* where the next input data can be read from. This index must be handled by the application and is only be read by the driver to check a FIFO overflow. The driver initializes this index to 0 when sequencer starts.

bufSize

Specifies the array size of buffer. This value must be the same as used for *s* in *TP851_CALC_SIZE_ADC_SEQDATA_BUF(s)* when calculating the size for the allocated buffer.

seqState

Displays the sequencer state. This is an ORed value of the following status flags.

flag	**description**
TP851_SF_SEQACTIVE	If set the ADC sequencer is started. If not set, the ADC sequencer stopped.
TP851_SF_SEQOVERFLOWERR	If set the ADC sequencer has detected an overflow error. (Hardware detected)
TP851_SF_SEQTIMERERROR	If set the ADC sequencer has detected a timer error. (Hardware detected)
TP851_SF_SEQIRAMERROR	If set the ADC sequencer has detected an instruction RAM error. (Hardware detected)
TP851_SF_SEQFIFOOVERFLOW	If set the application supplied FIFO (*buffer*) has overrun. Data got lost.

buffer

Array used for ADC sequencer data FIFO.
The ADC data is stored by the sequencer into this FIFO. The assignment from data to channel is done as follows. The first data will be from the lowest enabled channel, the second from the next enabled channel and so on. There will be no data stored for disabled channels. If the end of *buffer* is reached the next data will be stored again at the beginning of the buffer.

<u>Example:</u>
Enabled channels: 1, 2, 5
Buffer Size: 10
The table shows the index the data is stored to for channel and cycle.

sequencer cycle	**channel 1**	**channel 2**	**channel 5**
1st	0	1	2
2nd	3	4	5
3rd	6	7	8
4th	9	0	1
5th	2	3	4
…	…	…	…

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
OVERLAPPED                   AdcSeqOverlapped;
TP851_ADC_SEQSTART_BUF*      pAdcSeqStartBuf;
long                         realBufSize;

AdcSeqOverlapped.Offset = 0;
AdcSeqOverlapped.hEvent = 0;

/*
** allocate Buffer (100 word FIFO)
*/
realBufSize = TP851_CALC_SIZE_ADC_SEQDATA_BUF(100);
pAdcSeqStartBuf = (TP851_ADC_SEQSTART_BUF*)malloc(realBufSize);
pAdcSeqStartBuf ->bufSize = 100;

/*
** Start sequencer with a buffer of 100 words and a cycle time of 100 ms,
** do not use external trigger
*/
pAdcSeqStartBuf->cycTime     = 1000;
pAdcSeqStartBuf->flags       = 0;

printf("Start ADC Sequencer ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_ADC_SEQSTART,       // control code
     NULL,
     0,
     pAdcSeqStartBuf,                // pointer to buffer
     realBufSize,                    // size of buffer
     &NumBytes,
     &AdcSeqOverlapped               // overlapped I/O necessary
);

if( !success) {
     if (GetLastError() != ERROR_IO_PENDING)
          {
               /* handle error */
```

```
            } else {
            printf("OK\n");
        }
    }
```

Error Codes

ERROR_BUSY	The ADC sequencer is already running.
ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_INVALID_PARAMETER	Specified flags are invalid.

All other returned error codes are system error conditions.

4.1.3.4 IOCTL_TP851_ADC_SEQSTOP

This TPMC851 control function stops the ADC sequencer. All sequencer channel configurations remain valid after stopping. The parameters *lpInBuffer* and *lpOutBuffer* are not used and should be set to NULL.

Example

```
#include “TPMC851.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;

/*
** stop the ADC sequencer
*/
printf("Stop ADC sequencer ... ");
success = DeviceIoControl (
     hDevice,                       // TPMC851 handle
     IOCTL_TP851_ADC_SEQSTOP,       // control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                           // no overlapped I/O
);

if (success)
{
     printf("OK\n");
} else {
     /* handle error */
}
```

Error Codes

ERROR_ACCESS_DENIED	The sequencer is not running

All other returned error codes are system error conditions.

4.1.3.5 IOCTL_TP851_DAC_WRITE

This TPMC851 control function writes the specified value to the specified DAC channel. A pointer to the callers buffer (*TP851_DAC_WRITE_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

The DAC sequencer must be stopped to execute this function.

```
typedef struct
{
        int                     channel;
        unsigned long           flags;
        short                   dacValue;
} TP851_DAC_WRITE_BUF, *PTP851_DAC_WRITE_BUF;
```

channel

Specifies the DAC channel number. Valid values are 1..8.

flags

Is an ORed value of the following flags:

flag	**description**
TP851_F_CORR	If set the function will correct the *dacValue* before writing to DAC channel. Factory set and module dependent correction data is used for correction. If not set, *dacValue* is written to the DAC channel.
TP851_F_NOUPDATE	If set the DACs will not update after changing the DAC value. The output voltage will change with the next write with unset *TP851_F_NOUPDATE* flag. If not set the DAC will immediately convert and output the new voltage.

dacValue

This value is written to the DAC channel.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_DAC_WRITE_BUF     dacWriteBuf;

/*
** Write uncorrected 0x4000 to DAC channel 5
*/
dacWriteBuf.channel     = 5;
dacWriteBuf.flags       = 0;
dacWriteBuf.dacValue    = 0x4000;

printf("Write to DAC ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_DAC_WRITE,          // control code
     &dacWriteBuf,                   // pointer to buffer
     sizeof(TP851_DAC_WRITE_BUF),    // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                            // no overlapped I/O
);
if (success)
{
     printf("OK\n");
} else {
     /* handle error */
}
```

Error Codes

ERROR_BUSY	The DAC sequencer is currently running.
ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small.
ERROR_INVALID_PARAMETER	Specified flags are invalid.
ERROR_ACCESS_DENIED	Invalid channel number specified.

All other returned error codes are system error conditions.

4.1.3.6 IOCTL_TP851_DAC_SEQCONFIG

This TPMC851 control function configures and enables or simply disables a DAC channel for sequencer usage. A pointer to the callers buffer (*TP851_DAC_SEQCONFIG_BUF*) must be passed to the driver by the *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

The DAC sequencer must be stopped to execute this function.

```
typedef struct
{
        int                     channel;
        int                     enable;
        unsigned long           flags;
} TP851_DAC_SEQCONFIG_BUF, *PTP851_DAC_SEQCONFIG_BUF;
```

channel

Specifies the DAC channel number to configure. Valid values are 1..8.

enable

Specifies if the channel shall be used in sequencer mode or not. (0 disables the channel; any other value will enable the channel)

flags

Is an ORed value of the following flags:

flag	**description**
TP851_F_CORR	If set the function will correct the *dacValue* before writing to DAC channel. Factory set and module dependent correction data is used for correction. If not set, *dacValue* is written to the DAC channel.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_DAC_SEQCONFIG_BUF     dacSeqConfBuf;

/*
** Configure DAC channel 1, using corrected data
*/
adcSeqConfBuf.channel   = 1;
adcSeqConfBuf.enable    = TRUE;
adcSeqConfBuf.flags     = TP851_F_CORR;

printf("Configure channel for Sequencer ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_DAC_SEQCONFIG,      // control code
     &dacSeqConfBuf,                 // pointer to buffer
     sizeof(TP851_DAC_SEQCONFIG_BUF),// size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                            // no overlapped I/O
);
if (success)
{
     printf("OK\n");
} else {
     /* handle error */
}
```

Error Codes

ERROR_BUSY	The DAC sequencer is currently running.
ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_ACCESS_DENIED	Invalid channel number specified.
ERROR_INVALID_PARAMETER	Specified flags invalid.

All other returned error codes are system error conditions.

4.1.3.7 IOCTL_TP851_DAC_SEQSTART

This TPMC851 control function configures the DAC sequencer's cycle time and starts the DAC sequencer. A pointer to the callers buffer (*TP851_DAC_SEQSTART_BUF*) must be passed to the driver by *lpOutBuffer* parameter. *lpInBuffer* is not used and should be set to NULL. This function requires overlapped I/O.

```
typedef struct
{
        unsigned short          cycTime;
        unsigned long           flags;
        long                    putIdx;
        long                    getIdx;
        long                    bufSize;
        long                    seqState;
        short                   buffer[1];
} TP851_DAC_SEQSTART_BUF, *PTP851_DAC_SEQSTART_BUF;
```

cycTime

Specifies the DAC sequencer cycle time. The sequencer time is specified in 100µs steps. With a value of 0, the "Sequencer Continuous Mode" is selected.

flags

Is an ORed value of the following flags:

flag	description
TP851_F_EXTTRIGSRC	If set the DAC sequencer is trigger with digital I/O line 1. If not set, the DAC sequencer uses the DAC cycle counter.
TP851_F_EXTTRIGOUT	If set the DAC trigger is used as output on digital I/O line 1.
TP851_F_DACSEQREPEAT	If set the DAC will repeat data when the end of the buffer is reached, the *TP851_SF_SEQFIFOUNDERFLOW* error will be suppressed.

TP851_F_EXTTRIGSRC* and *TP851_F_EXTTRIGOUT cannot be used at the same time.

putIdx

Specifies the index into *buffer* where the next data shall be stored to. This index must be handled by the application and is only be read by the driver to check a FIFO underrun.

getIdx

Specifies the index into *buffer* where the next data will be read from. This index is handled by the driver and should only be read by the application to check if there is space for new data.

bufSize

Specifies the array size of buffer. This value must be the same as used for *s* in *TP851_CALC_SIZE_DAC_SEQDATA_BUF(s)* when calculating the size for the allocated buffer.

seqState

Displays the sequencer state. This is an ORed value of the following status flags.

flag	**description**
TP851_SF_SEQACTIVE	If set the DAC sequencer is started. If not set, the DAC sequencer stopped.
TP851_SF_SEQUNDERFLOWERR	If set the DAC sequencer has detected an underrun error. (Hardware detected)
TP851_SF_SEQFIFOUNDERFLOW	If set the application supplied FIFO (*buffer*) is empty and the sequencer could not write new data.

buffer

Array used for DAC sequencer data FIFO.
The DAC data is stored by the application into this FIFO. The assignment from data to channel is done as follows. The first data will be used for the lowest enabled channel, the second from the next enabled channel and so on. There will be no data used for disabled channels. If the end of *buffer* is reached the next data will be read again from the beginning of the buffer.

Example:
Enabled channels: 1, 2, 5
Buffer Size: 10
The table shows the index the data is used to for channel and cycle.

sequencer cycle	**channel 1**	**channel 2**	**channel 5**
1st	0	1	2
2nd	3	4	5
3rd	6	7	8
4th	9	0	1
5th	2	3	4
…	…	…	…

Example

```
#include "tpmc851.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
OVERLAPPED                  DacSeqOverlapped;
TP851_DAC_SEQSTART_BUF*     pDacSeqStartBuf;
long                        realBufSize;

DacSeqOverlapped.Offset = 0;
DacSeqOverlapped.hEvent = 0;

/*
** allocate Buffer (100 word FIFO)
*/
realBufSize = TP851_CALC_SIZE_DAC_SEQDATA_BUF(100);
pDacSeqStartBuf = (TP851_DAC_SEQSTART_BUF*)malloc(realBufSize);

pDacSeqStartBuf ->bufSize = 100;

/* Fill buffer */
seqBuf->buffer[0] = …;
seqBuf->buffer[1] = …;
seqBuf->buffer[2] = …;
…

/*
** Start sequencer with a buffer of 100 words and a cycle time of 100 ms,
** do not use external trigger
*/
pDacSeqStartBuf->cycTime    = 1000;
pDacSeqStartBuf->flags      = TP851_F_DACSEQREPEAT;

printf("Start DAC Sequencer ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_DAC_SEQSTART,       // control code
     NULL,
     0,
     pDacSeqStartBuf,                // pointer to buffer
     realBufSize,                    // size of buffer
     &NumBytes,
     &DacSeqOverlapped                // overlapped I/O necessary
);
if( !success) {
     if (GetLastError() != ERROR_IO_PENDING)
          {
               /* handle error */
          } else {
          printf("OK\n");
     }
}
```

Error Codes

ERROR_BUSY	The DAC sequencer is already running.
ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_INVALID_PARAMETER	Specified flags are invalid.

All other returned error codes are system error conditions.

4.1.3.8 IOCTL_TP851_DAC_SEQSTOP

This TPMC851 control function stops the DAC sequencer. All sequencer channel configurations remain valid after stopping. The parameters *lpInBuffer* and *lpOutBuffer* are not used and should be set to NULL.

Example

```
#include “TPMC851.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;

/*
** stop the DAC sequencer
*/
printf("Stop DAC sequencer ... ");
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_DAC_SEQSTOP,        // control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                            // no overlapped I/O
);

if (success)
{
     printf("OK\n");
} else {
     /* handle error */
}
```

Error Codes

ERROR_ACCESS_DENIED	The sequencer is not running

All other returned error codes are system error conditions.

4.1.3.9 IOCTL_TP851_IO_READ

This TPMC851 control function reads the current value of the digital I/O input lines. A pointer to the callers buffer (*TP851_IO_BUF*) must be passed to the driver by *lpOutBuffer* parameter. *lpInBuffer* is not used and should be set to NULL.

```
typedef struct
{
        unsigned short                          value;
} TP851_IO_BUF, *PTP851_IO_BUF;
```

value

Returns the current digital I/O input value.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_IO_BUF   ioBuf;

/*
** Read the digital I/O input value
*/
printf("Read I/O input value ... ");
success = DeviceIoControl (
     hDevice,                          // TPMC851 handle
     IOCTL_TP851_IO_READ,              // control code
     NULL,
     0,
     &ioBuf,                           // pointer to buffer
     sizeof(TP851_IO_BUF),             // size of buffer
     &NumBytes,
     NULL                              // no overlapped I/O
);
if (success)
{
     printf("OK\n    IO-value: 0x%04X\n", ioBuf.value);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small

All other returned error codes are system error conditions.

4.1.3.10 IOCTL_TP851_IO_WRITE

This TPMC851 control function writes a value to the digital I/O output lines. A pointer to the callers buffer (*TP851_IO_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
    unsigned short                value;
} TP851_IO_BUF, *PTP851_IO_BUF;
```

value

Specifies the new digital I/O output value.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_IO_BUF  ioBuf;

/*
** write 0x1234 to digital I/O output lines
*/
ioBuf.value = 0x1234;

printf("Write I/O output value ... ");
success = DeviceIoControl (
     hDevice,                          // TPMC851 handle
     IOCTL_TP851_IO_WRITE,             // control code
     &ioBuf,                           // pointer to buffer
     sizeof(TP851_IO_BUF),             // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                              // no overlapped I/O
);
if (success)
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small

All other returned error codes are system error conditions.

4.1.3.11 IOCTL_TP851_IO_EVENTWAIT

This TPMC851 control function waits for an event on a digital I/O input line. A pointer to the callers buffer (*TP851_IO_EVENTWAIT_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        int                     ioLine;
        unsigned long           flags;
        long                    timeout;
} TP851_IO_EVENTWAIT_BUF, *PTP851_IO_EVENTWAIT_BUF;
```

ioLine

Specifies the digital I/O line where the event shall occur. Valid values are 0..15.

flags

Specifies the event that shall occur. This is an ORed value of the following flags:

flag	**description**
TP851_F_HI2LOTRANS	If set, the function will return after a high to low transition occurs.
TP851_F_LO2HITRANS	If set, the function will return after a low to high transition occurs.

At least on flag must be specified.

timeout

Specifies the maximum time the function will wait for the specified event. The time is specified in seconds.

Example

```
#include "tpmc851.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_IO_EVENTWAIT_BUF  waitBuf;

/*
** Wait for a transition on I/O line 12 (max wait 10 seconds)
*/
waitBuf.ioLine =   12;
waitBuf.flags =    TP851_F_HI2LOTRANS | TP851_F_LO2HITRANS;
waitBuf.timeout =  10;

printf("Wait for I/O event ... ");
```

```
success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_IO_EVENTWAIT,       // control code
     &waitBuf,                       // pointer to buffer
     sizeof(TP851_IO_BUF),           // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                            // no overlapped I/O
);
if (success)
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_INVALID_PARAMETER	Specified flags are invalid.
ERROR_ACCESS_DENIED	Invalid channel number specified.
ERROR_NOT_ENOUGH_MEMORY	Too many wait events are queued.

All other returned error codes are system error conditions.

4.1.3.12 IOCTL_TP851_IO_CONFIG

This TPMC851 control function configures the digital I/O direction for each line. A pointer to the callers buffer (*TP851_IO_CONF_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        unsigned short              direction;
} TP851_IO_CONF_BUF, *PTP851_IO_CONF_BUF;
```

direction

Specifies the new direction setting for the digital I/O lines. A bit set to 1 enables output, a 0 means that the I/O line is input.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_IO_CONF_BUF  ioConfBuf;

/*
** Enable lines 0,2,8,9 for output, all other lines are input
*/
ioConfBuf.direction = (1 << 0) | (1 << 2) | (1 << 8) | (1 << 9);

printf("Set new I/O configuration ... ");
success = DeviceIoControl (
     hDevice,                          // TPMC851 handle
     IOCTL_TP851_IO_CONFIG,            // control code
     &ioConfBuf,                       // pointer to buffer
     sizeof(TP851_IO_CONF_BUF),        // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                              // no overlapped I/O
);
if (success)
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small

All other returned error codes are system error conditions.

4.1.3.13 IOCTL_TP851_IO_DEBCONFIG

This TPMC851 control function configures the digital I/O input debouncer circuit. A pointer to the callers buffer (*TP851_IO_DEBCONF_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        unsigned short              enableMask;
        unsigned short              debTime;
} TP851_IO_DEBCONF_BUF, *PTP851_IO_DEBCONF_BUF;
```

enableMask

Specifies digital I/O lines which shall be used with debouncer. A bit set to 1 enables the debouncer, and a 0 disables the debouncer for the specific I/O line.

debTime

Specifies the debounce time. The time is specified in 100ns steps.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_IO_DEBCONF_BUF    ioDebConfBuf;

/*
** Enable Debouncer for line 0 and 2 (debounce time 1ms)
*/
ioDebConfBuf.enableMask =    (1 << 0) | (1 << 2);
ioDebConfBuf.debTime =       10000;

printf("Set debouncer  configuration ... ");

success = DeviceIoControl (
     hDevice,                           // TPMC851 handle
     IOCTL_TP851_IO_DEBCONFIG,          // control code
     &ioDebConfBuf,                     // pointer to buffer
     sizeof(TP851_IO_DEBCONF_BUF),      // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                               // no overlapped I/O
);
if (success)
```

```
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small

All other returned error codes are system error conditions.

4.1.3.14 IOCTL_TP851_CNT_READ

This TPMC851 control function reads the current value of the counter/timer. A pointer to the callers buffer (*TP851_CNT_READ_BUF*) must be passed to the driver by *lpOutBuffer* parameter. *lpInBuffer* is not used and should be set to NULL.

```
typedef struct
{
        unsigned long           count;
        unsigned long           state;
} TP851_CNT_READ_BUF, *PTP851_CNT_READ_BUF;
```

count

Returns the actual counter value.

state

Returns the counter state. If possible the flags are cleared after read. This is an ORed value of the following flags.

flag	**description**
TP851_SF_CNTBORROW	Counter borrow bit set (actual state)
TP851_SF_CNTCARRY	Counter carry bit set (actual state)
TP851_SF_CNTMATCH	Counter match event has occurred since last read.
TP851_SF_CNTSIGN	Counter sign bit (actual state)
TP851_SF_CNTDIRECTION	If set, counter direction is upward. If not set, counter direction is downward.
TP851_SF_CNTLATCH	Counter value has been latched.
TP851_SF_CNTLATCHOVERFLOW	Counter latch overflow has occurred.
TP851_SF_CNTSNGLCYC	Counter Single Cycle is active

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_CNT_READ_BUF cntBuf;

/*
** Read the counter/timer value
*/
printf("Read counter ... ");

success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
     IOCTL_TP851_CNT_READ,           // control code
     NULL,
     0,
     &cntBuf,                        // pointer to buffer
     sizeof(TP851_CNT_READ_BUF),     // size of buffer
     &NumBytes,
     NULL                            // no overlapped I/O
);
if (success)
{
     printf("OK\n);
     printf("    Counter: %ld", cntBuf.counter);
     printf("    State:   %lXh", cntBuf.state);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small

All other returned error codes are system error conditions.

4.1.3.15 IOCTL_TP851_CNT_MATCHWAIT

This TPMC851 control function waits for the counter match event to occur. A pointer to the callers buffer (*TP851_CNT_WAIT_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        long                    timeout;
} TP851_CNT_WAIT_BUF, *PTP851_CNT_WAIT_BUF;
```

timeout

Specifies the maximum time the function will wait for the specified event. The time is specified in seconds.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_CNT_WAIT_BUF     waitBuf;

/*
** Wait for counter match event (max wait 10 seconds)
*/
waitBuf.timeout =  10;

printf("Wait for counter match event ... ");
success = DeviceIoControl (
     hDevice,                           // TPMC851 handle
     IOCTL_TP851_CNT_MATCHWAIT,         // control code
     &waitBuf,                          // pointer to buffer
     sizeof(TP851_CNT_WAIT_BUF),        // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                               // no overlapped I/O
);
if (success)
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_NOT_ENOUGH_MEMORY	Too many wait events are queued.

All other returned error codes are system error conditions.

4.1.3.16 IOCTL_TP851_CNT_CTRLWAIT

This TPMC851 control function waits for a counter control event to occur. The specific event is chosen by a call to Ioctl function IOCTL_TP851_CNT_CONFIG. A pointer to the callers buffer (*TP851_CNT_WAIT_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        long                    timeout;
} TP851_CNT_WAIT_BUF, *PTP851_CNT_WAIT_BUF;
```

timeout

Specifies the maximum time the function will wait for the specified event. The time is specified in seconds.

Example

```
#include  “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_CNT_WAIT_BUF      waitBuf;

/*
** Wait for counter control event (max wait 10 seconds)
*/
waitBuf.timeout =  10;
printf("Wait for counter match event ... ");
success = DeviceIoControl (
     hDevice,                           // TPMC851 handle
     IOCTL_TP851_CNT_CTRLWAIT,          // control code
     &waitBuf,                          // pointer to buffer
     sizeof(TP851_CNT_WAIT_BUF),        // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                               // no overlapped I/O
);
if (success)
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_NOT_ENOUGH_MEMORY	Too many wait events are queued.

All other returned error codes are system error conditions.

4.1.3.17 IOCTL_TP851_CNT_CONFIG

This TPMC851 control function configures the counter. A pointer to the callers buffer (*TP851__CNT_CONFIG_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        unsigned long           inputMode;
        int                     clockDivider;
        unsigned long           countMode;
        unsigned long           controlMode;
        unsigned long           invFlags;
} TP851_CNT_CONFIG_BUF, *PTP851_CNT_CONFIG_BUF;
```

inputMode

Specifies the counter input mode. The following modes are defined and valid:

flag	**description**
TP851_M_CNTIN_DISABLE	Counter disabled
TP851_M_CNTIN_TIMERUP	Timer Mode Up
TP851_M_CNTIN_TIMERDOWN	Timer Mode Down
TP851_M_CNTIN_DIRCOUNT	Direction Count
TP851_M_CNTIN_UPDOWNCOUNT	Up/Down Count
TP851_M_CNTIN_QUAD1X	Quadrature Count 1x
TP851_M_CNTIN_QUAD2X	Quadrature Count 2x
TP851_M_CNTIN_QUAD3X	Quadrature Count 4x

clockDivider

Specifies clock divider. Allowed clock divider values are 1 (40MHz), 2 (20MHz), 4 (10MHz) and 8 (5MHz).

countMode

Specifies the count mode. The following modes are defined and valid:

flag	**description**
TP851_M_CNT_CYCLE	Cycling Counter
TP851_M_CNT_DIVN	Divide-by-N
TP851_M_CNT_SINGLE	Single Cycle

controlMode

Specifies the counter control mode. These events can generate counter control events. The following modes are defined and valid:

flag	description
TP851_M_CNTCTRL_NONE	No Control Mode
TP851_M_CNTCTRL_LOAD	Load Mode
TP851_M_CNTCTRL_LATCH	Latch Mode
TP851_M_CNTCTRL_GATE	Gate Mode
TP851_M_CNTCTRL_RESET	Reset Mode

invFlags

Specifies if counter input lines shall be inverted or not. This is an ORed value of the following flags:

flag	description
TP851_F_CNTINVINP2	If set, input line 2 is low active If not set, input line 2 is high active
TP851_F_CNTINVINP3	If set, input line 3 is low active If not set, input line 3 is high active
TP851_F_CNTINVINP4	If set, input line 4 is low active If not set, input line 4 is high active

Example

```
#include “tpmc851.h”

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_CNT_CONFIG_BUF    cntConfBuf;

/*
** Setup counter for direction count, clock divider 1, cycling count,
** no control mode and all line high active
*/
cntConfBuf. inputMode =     TP851_M_CNTIN_DIRCOUNT;
cntConfBuf. clockDivider =  1;
cntConfBuf. countMode =     TP851_M_CNT_CYCLE;
cntConfBuf. controlMode =   TP851_M_CNTCTRL_NONE;
cntConfBuf. invFlags =      0;

printf("Set counter configuration ... ");

success = DeviceIoControl (
     hDevice,                        // TPMC851 handle
```

```
        IOCTL_TP851_CNT_CONFIG,           // control code
        &cntConfigBuf,                    // pointer to buffer
        sizeof(TP851_CNT_CONFIG_BUF),     // size of buffer
        NULL,
        0,
        &NumBytes,
        NULL                              // no overlapped I/O
);
if (success)
{
        printf("OK\n);
} else {
        /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_INVALID_PARAMETER	Specified parameters are invalid.

All other returned error codes are system error conditions.

4.1.3.18 IOCTL_TP851_CNT_RESET

This TPMC851 control function resets the counter value. The parameters *lpInBuffer* and *lpOutBuffer* are not used and should be set to NULL.

Example

```
#include “TPMC851.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;

/*
** reset the counter value
*/
printf("Reset counter value ... ");
success = DeviceIoControl (
     hDevice,                         // TPMC851 handle
     IOCTL_TP851_CNT_RESET,           // control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,
     NULL                             // no overlapped I/O
);

if (success)
{
     printf("OK\n");
} else {
     /* handle error */
}
```

Error Codes

All returned error codes are system error conditions.

4.1.3.19 IOCTL_TP851_CNT_SETPRELD

This TPMC851 control function sets the counter's preload value. A pointer to the callers buffer (*TP851_CNT_SETPRELD_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
    unsigned long                value;
    unsigned long                flags;
} TP851_CNT_SETPRELD_BUF, *PTP851_CNT_SETPRELD_BUF;
```

value

Specifies the new counter preload value.

flags

Is an ORed value of the following flags:

flag	**description**
TP851_F_IMMPRELOAD	If set, the function will immediate load the preload value into the counter If not set, preload value will be used for the next preload condition.

Example

```
#include "tpmc851.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_CNT_SETPRELD_BUF  cntPrldBuf;

/*
** Immediately load 0x11223344 into the counter and preload register
*/
cntPrldBuf. value       = 0x11223344;
cntPrldBuf.flags        = TP851_F_IMMPRELOAD;

printf("Set preload value ... ");
success = DeviceIoControl (
    hDevice,                          // TPMC851 handle
    IOCTL_TP851_CNT_SETPRELD,         // control code
    &cntPrldBuf,                      // pointer to buffer
    sizeof(TP851_CNT_SETPRELD_BUF), // size of buffer
    NULL,
    0,
    &NumBytes,
    NULL                             // no overlapped I/O
);
if (success)
{
    printf("OK\n);
} else {
    /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small
ERROR_INVALID_PARAMETER	Specified flags are invalid.

All other returned error codes are system error conditions.

4.1.3.20 IOCTL_TP851_CNT_SETMATCH

This TPMC851 control function sets the counter's match value. If the counter value is the same as the match value, an event will occur. The driver can be used to wait for this event with Ioctl function IOCTL_TP851_CNT_MATCHWAIT. A pointer to the callers buffer (*TP851_CNT_SETMATCH_BUF*) must be passed to the driver by *lpInBuffer* parameter. *lpOutBuffer* is not used and should be set to NULL.

```
typedef struct
{
        unsigned long           value;
} TP851_CNT_SETMATCH_BUF, *PTP851_CNT_SETMATCH_BUF;
```

value

Specifies the new counter match value.

Example

```
#include  "tpmc851.h"

HANDLE hDevice;
BOOLEAN success;
ULONG NumBytes;
TP851_CNT_SETMATCH_BUF  cntMatchBuf;

/*
** Set match value to 0x10000
*/
cntMatchBuf.value       = 0x10000;
printf("Set match value ... ");
success = DeviceIoControl (
     hDevice,                           // TPMC851 handle
     IOCTL_ TP851_CNT_SETMATCH,         // control code
     &cntMatchBuf,                      // pointer to buffer
     sizeof(TP851_CNT_SETMATCH_BUF),    // size of buffer
     NULL,
     0,
     &NumBytes,
     NULL                               // no overlapped I/O
);
if (success)
{
     printf("OK\n);
} else {
     /* handle error */
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The size of the provided buffer is too small

All other returned error codes are system error conditions.