The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC851-SW-65

Windows Device Driver

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

Version 3.1.x

User Manual

Issue 3.1.0

May 2017

powerBridge Computer

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

www.powerbridge.de
info@powerbridge.de

EWS TECHNOLOGIES GmbH

m Bahnhof 7 25469 Halstenbek, Germany
4101 4058 0 Fax: +49 (0) 4101 4058 19
o@tews.com www.tews.com

TPMC851-SW-65

Windows Device Driver

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

Supported Modules:
TPMC851

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	File list modified (subdirectory)	August 5, 2008
2.0.0	Driver supports Windows7, Description of API Functions added, Description of I/O Functions removed, General Revision	October 10, 2011
2.1.0	Examples error handling and error codes corrected	June 12, 2013
2.2.0	Chapter Installation removed, replaced by Installation Guide, Correction quadrature count configuration names	October 27, 2014
3.0.0	tpmc851AdcSeqStart() and tpmc851DacSeqStart() modified tpmc851AdcSeqRead() and tpmc851DacSeqWrite() added	June 7, 2016
3.1.0	Support for managed code added (tpmc851 class)	May 3, 2017

Table of Contents

1 Introduction

The TPMC851-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.

The TPMC851-SW-65 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

The TPMC851-SW-65 device driver supports the modules listed below:

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

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

TPMC851 User Manual
Installation-Guide on Distribution Media

2 Installation

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

driver*	Directory containing driver files
example\tpmc851exa.c	Example application source
api\tpmc851api.h	Application Programming Interface header
TPMC851-SW-65-3.1.0.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history
tpmc851exa.exe	Example application (executable for quick start)

Copy all required Header and API files into your desired project directory.

For device driver installation execute the application “Setup” on the distribution media. For more information refer to the Installation-Guide, which is also part of the distribution media.

3 API Documentation

3.1 Usage of Function and Class Interface

3.1.1 API Function Interface

To use the standard API-function interface in non-managed applications, the tpmc851api.dll must be used.

3.1.2 Class Interface

The class interface offers a simple way to use the API interface in managed code application.

To use this class, it is necessary that tpmc851class.dll is available for the application and to reference the namespace "Tews.Tpmc851".

Constructor and Destructor

The object constructor is equivalent to the tpmc851Open function and the destructor is equivalent to the tpmc851Close function.

Class Constants

TPMC851 definitions and flags are defined as constants in tpmc851 class. The names of the constants in the class are matching the names of the API, except the leading "TPMC851_" is removed. See the examples below:

API	Class
TPMC851_OK	tpmc851.OK
TPMC851_ERR_INVALID_HANDLE	tpmc851.ERR_INVALID_HANDLE
TPMC851_SF_SEQACTIVE	tpmc851.SF_SEQACTIVE
TPMC851_M_CNTCTRL_NONE	tpmc851.M_CNTCTRL_NONE

Class Methods

TPMC851 functions are implemented as methods in tpmc851 class.

Generally all method parameters can be used as they are used in the API. The types and constants of the class must be used instead. The name of the class methods matches the function names of the API, except the leading "tpmc851" is removed.

The parameter hdl of the API is assigned implicitly to the object. So there is no hdl parameter for class methods.

C# Example

```
…

using Tews.Tpmc851;

…

private tpmc851 tpmc851Dev;

…

// Open TPMC851 device
try
{
     tpmc851Dev = new tpmc851("\\\\.\\TPMC851_1");
}
catch
{
     // Error handling (No matching device found)
}

…

// ----------- Start sequencer execution -----------
// * cycle time: 100ms
// * do not use external trigger signals
// * buffer size: 1000 (1000 cycles)
UInt32 retVal = tpmc851Dev.AdcSeqStart(1000, 0, 1000);
if (retVal != tpmc851.OK)
{
     // Error handling (Sequencer Start/Setup failed)
}

…
```

```
…

// Read from Sequencer loop
UInt32 retVal = 0;
Int32[] inpVal = new Int32[32];
UInt32 status = 0;

while (…)
{
     // Read next data from ADC sequencer buffer
     retVal = tpmc851Dev.AdcSeqRead(0, ref inpVal, ref status);

     if (retVal != tpmc851.OK)
     {
          if (retVal != tpmc851.ERR_NODATA)
          {
               // Error handling (Sequencer Read failed)
          }
          else
          {
               // No data available
          }
     }
     else
     {
          // Valid data read
     }

     // Delay
}

…

UInt32 retVal = tpmc851Dev.AdcSeqStop();
if (retVal != tpmc851.OK)
{
     // Error handling (Sequencer Stop failed)
}
```

3.2 General Functions

3.2.1 tpmc851Open

NAME

tpmc851Open – Opens a Device

SYNOPSIS

<u>API-function:</u>

```
TPMC851_HANDLE tpmc851Open
(
        char   *DeviceName
)
```

<u>Class-constructor:</u>

```
tpmc851
(
        String DevName
)
```

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 “tpmc851api.h”

TPMC851_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tpmc851Open(“\\\\.\\TPMC851_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.

3.2.2 tpmc851Close

NAME

tpmc851Close – Closes a Device

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851Close
(
        TPMC851_HANDLE        hdl
)
```

Class-destructor

~tpmc851()

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 "tpmc851api.h"

TPMC851_HANDLE hdl;
TPMC851_STATUS result;

/*
** close file descriptor to device
*/
result = tpmc851Close(hdl);
if (result != TPMC851_OK)
{
     /* handle close error */
}
```

RETURNS

On success TPMC851_OK, or an appropriate error code.

ERROR CODES

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

3.3 Device Access Interface

3.3.1 tpmc851AdcRead

NAME

tpmc851AdcRead – Read value from ADC channel

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851AdcRead
(
        TPMC851_HANDLE          hdl,
        int                     channel,
        int                     gain,
        unsigned int            flags,
        short                   *pAdcValue
)
```

Class-method:

```
UInt32 AdcRead
(
        Int32                   channel,
        Int32                   gain,
        UInt32                  flags,
        ref Int16               pAdcValue
)
```

DESCRIPTION

This function starts an ADC conversion with specified parameters, waits for completion and returns the value.

The ADC sequencer must be stopped for single ADC conversions.

PARAMETERS

hdl

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

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

This is an ORed value of the following flags:

Flag	**Description**
TPMC851_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*.
TPMC851_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.
TPMC851_F_DIFF	If set the input channel will be a differential input. If not set the input channel will be a single-ended input.

pAdcValue

This parameter specifies a pointer to a *short* value which receives the current ADC value.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;
short              AdcValue;

…
```

…

```
/*--------------------------------------------------------------------
  Read a corrected value from differential channel 2 using gain of 4
  --------------------------------------------------------------------*/
result = tpmc851AdcRead(
              hdl,
              2,                                /* Channel    */
              4,                                /* Gain       */
              TPMC851_F_CORR | TPMC851_F_DIFF,  /* Flags      */
              &AdcValue );                      /* ADC value  */

if (result == TPMC851_OK)
{
     /* function succeeded */
     printf("    ADC-value: %d", AdcValue);
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	The ADC sequencer is currently running
TPMC851_ERR_INVAL	Invalid flags, gain value, or buffer specified
TPMC851_ERR_ACCESS	Invalid ADC channel number specified
TPMC851_ERR_TIMEOUT	ADC conversion timed out

3.3.2 tpmc851AdcSeqConfig

NAME

tpmc851AdcSeqConfig – Configure ADC sequencer channel

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851AdcSeqConfig
(
        TPMC851_HANDLE          hdl,
        int                     channel,
        int                     enable,
        int                     gain,
        unsigned int            flags
)
```

Class-method:

```
UInt32 AdcSeqConfig
(
        Int32                   channel,
        Boolean                 enable,
        Int32                   gain,
        UInt32                  flags
)
```

DESCRIPTION

This function enables and configures, or disables an ADC channel for sequencer use.

The ADC sequencer must be stopped to execute this function.

PARAMETERS

hdl

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

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
TPMC851_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.
TPMC851_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 "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*--------------------------------------------------------
  Configure single-ended channel 3, using a gain of 4 and
  returning corrected data when the sequencer is running
  --------------------------------------------------------*/
result = tpmc851AdcSeqConfig(
              hdl,
              3,                              /* Channel    */
              1,                              /* Enable     */
              4,                              /* Gain       */
              TPMC851_F_CORR );               /* Flags      */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flags or gain value specified
TPMC851_ERR_ACCESS	Invalid ADC channel number specified
TPMC851_ERR_BUSY	The ADC sequencer is currently running

3.3.3 tpmc851AdcSeqStart

NAME

tpmc851AdcSeqStart – Start ADC Sequencer

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851AdcSeqStart
(
        TPMC851_HANDLE          hdl,
        unsigned short          cycTime,
        unsigned int            flags,
        unsigned int            NumOfBufferPages
)
```

Class-method:

```
UInt32 AdcSeqStart
(
        UInt16                  cycTime,
        UInt32                  flags,
        UInt32                  NumOfBufferPages
)
```

DESCRIPTION

This function configures the ADC sequencer time and starts the ADC sequencer.

PARAMETERS

hdl

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

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

One of the following optional flags is possible:

Flag	Description
TPMC851_F_EXTTRIGSRC	If set the ADC sequencer is triggered with digital I/O line 0. If not set, the ADC sequencer uses the ADC cycle counter.
TPMC851_F_EXTTRIGOUT	If set the ADC trigger is used as output on digital I/O line 0.

NumOfBufferPages

This parameter defines the size of the sequencers read buffer. It defines number of data sets which can be stored. A data set means one ADC-value per channel which is enabled in sequencer mode. For example if the parameters value is 100, data of 100 sequencer cycles can be stored.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE              hdl;
TPMC851_STATUS              result;

/*---------------------------------------------------
  Start sequencer with a buffer of 100 words and
  a cycle time of 100 ms, do not use external trigger
  and allocate Buffer for 100 sequencer cycles
  ---------------------------------------------------*/
result = tpmc851AdcSeqStart(
              hdl,
              1000,                     /* Cycle Time (in 100us)    */
              0,                        /* Flags                    */
              100 );                    /* sequencer buffer size    */
if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	The ADC sequencer is currently running
TPMC851_ERR_NOBUFS	Invalid buffer space
TPMC851_ERR_INVAL	Invalid flag, or buffer specified
TPMC851_ERR_BAD_FUNCTION	Starting overlapped I/O function failed

3.3.4 tpmc851AdcSeqRead

NAME

tpmc851AdcSeqRead – Read ADC Sequencer Data

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851AdcSeqRead
(
        TPMC851_HANDLE        hdl,
        unsigned int          flags,
        int                   *pData,
        unsigned int          *pStatus
)
```

Class-method:

```
UInt32 AdcSeqRead
(
        UInt32                flags,
        ref Int32[]           pData,
        ref UInt32            pStatus
)
```

DESCRIPTION

This function reads data of one sequencer cycle from the ADC sequencer buffer. The function returns immediately.

PARAMETERS

hdl

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

flags

One of the following optional flags is possible:

Flag	Description
TPMC851_F_FLUSH	If set the function will return the data of the latest sequencer cycle. If not set, the function returns the data of the oldest available sequencer cycle.

pData

This is a pointer to a data array where the ADC data of the read cycle will be stored. The array must be an int array with 32 elements. The data of ADC 1 will be stored to index 0, the data of ADC 2 will be stored to index 1, and so on.
All channels which are not included in the sequencer configuration will return 0.

pStatus

This is a pointer to an unsigned int value where the sequencer status will be returned. The status is an ORed value of the following flags:

Flag	Description
TPMC851_SF_SEQACTIVE	ADC sequencer mode is active
TPMC851_SF_SEQFIFOOVERFLOW	ADC sequencer FIFO overflow error occurred. The SW-FIFO is full and data was lost.
TPMC851_SF_SEQOVERFLOWERROR	ADC sequencer overflow error has been detected by hardware. The sequencer was stopped.
TPMC851_SF_SEQTIMERERROR	ADC sequencer timer configuration is invalid. The sequencer was stopped.
TPMC851_SF_SEQIRAMERROR	An invalid sequencer configuration has been detected by the hardware, e.g. no channel has been configured. The sequencer was stopped.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE          hdl;
unsigned int            flags;
int                     adcSeqBuf[32];
unsigned int            adcSeqStat;

/*----------------------------------------------------
  read ADC sequencer data of the latest cycle
  ----------------------------------------------------*/
result = tpmc851AdcSeqRead(
              hdl,
              TPMC851_F_FLUSH,        /* Flags: Read latest data   */
              adcSeqBuf,
              &adcSeqStat);           /* status buffer             */
if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_NOMEM	No sequencer buffer available, sequencer not started.
TPMC851_ERR_NODATA	No data available
TPMC851_ERR_INVAL	Invalid flag, or buffer specified

3.3.5 tpmc851AdcSeqStop

NAME

tpmc851AdcSeqStop – Stop ADC Sequencer

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851AdcSeqStop
(
        TPMC851_HANDLE        hdl
)
```

Class-method:

```
UInt32 AdcSeqStop()
```

DESCRIPTION

This function stops the ADC sequencer. All sequencer channel configurations remain valid after stopping.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

…

```
/*------------------
  Stop ADC sequencer
  ------------------*/
result = tpmc851AdcSeqStop( hdl );

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_ACCESS	The sequencer is not running

3.3.6 tpmc851DacWrite

NAME

tpmc851DacWrite – Write to DAC channel

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851DacWrite
(
        TPMC851_HANDLE          hdl,
        int                     channel,
        unsigned int            flags,
        short                   DacValue
)
```

Class-method:

```
UInt32 DacWrite
(
        Int32                   channel,
        UInt32                  flags,
        Int16                   DacValue
)
```

DESCRIPTION

This function writes a value to the DAC register and starts the conversion if specified.

The DAC sequencer must be stopped for single DAC writes.

PARAMETERS

hdl

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

channel

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

flags

Is an ORed value of the following flags:

Flag	**Description**
TPMC851_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.
TPMC851_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 *TPMC851_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 "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*------------------------------------------------------------
  Write uncorrected 0x4000 to DAC channel 5, immediate convert
  ------------------------------------------------------------*/
result = tpmc851DacWrite(
              hdl,
              5,                                /* Channel    */
              0,                                /* Flags      */
              0x4000 );                         /* DAC value  */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified
TPMC851_ERR_ACCESS	Invalid DAC channel number specified

3.3.7 tpmc851DacSeqConfig

NAME

tpmc851DacSeqConfig – Configure DAC sequencer channel

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851DacSeqConfig
(
        TPMC851_HANDLE         hdl,
        int                    channel,
        int                    enable,
        unsigned int           flags
)
```

Class-method:

```
UInt32 DacSeqConfig
(
        Int32                  channel,
        Boolean                enable,
        UInt32                 flags
)
```

DESCRIPTION

This function enables and configures, or disables a DAC channel for sequencer use.

The DAC sequencer must be stopped to execute this function.

PARAMETERS

hdl

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

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

The following optional flag is possible:

Flag	Description
TPMC851_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 "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*----------------------------------------------------
  Configure DAC channel 1, using corrected data while
  the sequencer is running
  ----------------------------------------------------*/
result = tpmc851DacSeqConfig(
              hdl,
              1,                                /* Channel    */
              1,                                /* Enable     */
              TPMC851_F_CORR );                 /* Flags      */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified
TPMC851_ERR_ACCESS	Invalid DAC channel number specified

3.3.8 tpmc851DacSeqStart

NAME

tpmc851DacSeqStart – Start DAC Sequencer

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851DacSeqStart
(
        TPMC851_HANDLE          hdl,
        unsigned short          cycTime,
        unsigned int            flags,
        unsigned int            NumOfBufferPages,
        unsigned int            NumDataPages,
        int                     *pData
)
```

Class-method:

```
UInt32 DacSeqStart
(
        UInt16                  cycTime,
        UInt32                  flags,
        UInt32                  NumOfBufferPages,
        UInt32                  NumDataPages,
        Int32[,]                pData
)
```

DESCRIPTION

This function configures the DAC sequencer time and starts the DAC sequencer.

PARAMETERS

hdl

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

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**
TPMC851_F_EXTTRIGSRC	If set the DAC sequencer is triggered with digital I/O line 1. If not set, the DAC sequencer uses the DAC cycle counter.
TPMC851_F_EXTTRIGOUT	If set the DAC trigger is used as output on digital I/O line 1.
TPMC851_F_DACSEQREPEAT	If set the DAC will repeat data when the end of the buffer is reached, the error *TPMC851_SF_SEQFIFOUNDERFLOW* is suppressed.

***TPMC851_F_EXTTRIGSRC* and *TPMC851_F_EXTTRIGOUT* cannot be used at the same time.**

NumOfBufferPages

This parameter defines the size of the sequencer's write buffer. It defines the number of data sets which shall be transferred into the DAC sequencer FIFO. A data set means one DAC-value per channel which is enabled in sequencer mode. For example if the parameters value is 100, data for 100 sequencer cycles can be stored.

NumDataPages

This parameter specifies data for how many DAC sequencer cycles are available in pData.

pData

This parameter is a pointer to a data buffer which contains DAC data for a number of DAC sequencer cycles. This data will be transferred into the driver's DAC sequencer FIFO before starting the sequencer.
The array must be a simple array, with an int value for every DAC channel for every sequencer cycle that shall be transferred. The data is assigned as follows: Data for the first cycle is stored at index 0, Data for the second cycle is stored at index 8, and so on. Data for DAC 1 will be stored with an index offset of 0, data for DAC 2 will be stored with an index offset of 1, and so on.
Values for channels not used in sequencer mode will be ignored.

Example:
Enabled channels: 1, 2, 5
NumOfPages: 4
The table shows the buffer index and the corresponding channel and data set (output cycle).

DAC-sequencer cycle	**DAC 1**	**DAC 2**	**DAC 3**	**DAC 4**	**DAC 5**	**DAC 6**	**DAC 7**	**DAC 8**
1st	0	1	2*)	3*)	4	5*)	6*)	7*)
2nd	8	9	10*)	11*)	12	13*)	14*)	15*)
3rd	16	17	18*)	19*)	20	21*)	22*)	23*)
4th	24	25	26*)	27*)	28	29*)	30*)	31*)

*) Data will not be used in this example configuration

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE              hdl;
TPMC851_STATUS              result;
int                         seqData[8*4];      /* (8 DACs) 4 data sets */

/*-----------------------------------------
  Fill buffer
  3 channels are enabled for sequencer mode
  -----------------------------------------*/
/* 1st Data set */
seqData[0] = …;    /* DAC 1 */
seqData[1] = …;    /* DAC 2 */
seqData[4] = …;    /* DAC 5 */

/* 2nd Data set */
seqData[8] = …;    /* DAC 1 */
seqData[9] = …;    /* DAC 2 */
seqData[12] = …;   /* DAC 5 */

/* 3rd Data set */
seqData[16] = …;   /* DAC 1 */
seqData[17] = …;   /* DAC 2 */
seqData[20] = …;   /* DAC 5 */

/* 4th Data set */
seqData[24] = …;   /* DAC 1 */
seqData[25] = …;   /* DAC 2 */
seqData[28] = …;   /* DAC 5 */

…
```

…

```
/*-----------------------------------------------------------------
  Start sequencer with a buffer of 100 words and
  a cycle time of 100 ms, do not use external trigger, repeat data

  3 channels are enabled for sequencer mode
  -----------------------------------------------------------------*/
result = tpmc851DacSeqStart(
              hdl,
              1000,                      /* Cycle Time (in 100us)    */
              100,
              4,
              seqData);                  /* Sequencer output data    */
if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_BUSY	The DAC sequencer is currently running
TPMC851_ERR_NOBUFS	Invalid buffer space
TPMC851_ERR_INVAL	Invalid flag, or buffer specified
TPMC851_ERR_BAD_FUNCTION	Starting overlapped I/O function failed

3.3.9 tpmc851DacSeqWrite

NAME

tpmc851DacSeqWrite – Transfer DAC Sequencer data into the sequencer buffer

SYNOPSIS

<u>API-function:</u>

```
TPMC851_STATUS tpmc851DacSeqWrite
(
        TPMC851_HANDLE          hdl,
        unsigned int            flags,
        unsigned int            *NumDataPages,
        int                     *pData,
        int                     *pStatus
)
```

<u>Class-method:</u>

```
UInt32 DacSeqWrite
(
        UInt32                  flags,
        ref UInt32              NumDataPages,
        Int32[,]                pData,
        ref UInt32              pStatus
)
```

DESCRIPTION

This function transfers data into the DAC sequencer buffer.

PARAMETERS

hdl

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

flags

This parameter is unused.

NumDataPages

This pointer to an unsigned int value specifies how many DAC sequencer cycles are available in pData. The function returns the number of successfully transferred data pages through this pointer.

pData

This parameter is a pointer to a data buffer which contains DAC data for a number of DAC sequencer cycles. This data will be transferred into the drivers DAC sequencer FIFO.
The array must be a simple array, with an int value for every DAC channel for every sequencer cycle that shall be transferred. The data is assigned as follows: Data for the first cycle is stored at index 0, Data for the second cycle is stored at index 8, and so on. Data for DAC 1 will be stored with an index offset of 0, data for DAC 2 will be stored with an index offset of 1, and so on.
Values for channels not used in sequencer mode will be ignored.

Example:
Enabled channels: 1, 2, 5
NumOfPages: 4
The table shows the buffer index and the corresponding channel and data cycle.

DAC-sequencer cycle	DAC 1	DAC 2	DAC 3	DAC 4	DAC 5	DAC 6	DAC 7	DAC 8
1st	0	1	2*)	3*)	4	5*)	6*)	7*)
2nd	8	9	10*)	11*)	12	13*)	14*)	15*)
3rd	16	17	18*)	19*)	20	21*)	22*)	23*)
4th	24	25	26*)	27*)	28	29*)	30*)	31*)

*) Data will not be used in this example configuration

pStatus

This is a pointer to an unsigned int value where the sequencer status will be returned to. The status is an ORed value of the following flags:

Flag	Description
TPMC851_SF_SEQACTIVE	ADC sequencer mode is active
TPMC851_SF_SEQFIFOUNDERFLOW	ADC sequencer FIFO underflow error occurred. The SW-FIFO is empty and old data will used.
TPMC851_SF_SEQUNDERFLOWERROR	ADC sequencer underflow error has been detected by hardware. The Sequencer uses the previous data values.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE          hdl;
TPMC851_STATUS          result;
unsigned int            NumDataPages;
int                     seqStatus;
int                     seqData[8*4];       /* (8 DACs) 4 data sets */

…
```

```
…
/*-----------------------------------------
  Fill buffer
  3 channels are enabled for sequencer mode
  -----------------------------------------*/
/* 1st Data set */
seqData[0] = …;    /* DAC 1 */
seqData[1] = …;    /* DAC 2 */
seqData[4] = …;    /* DAC 5 */

/* 2nd Data set */
seqData[8] = …;    /* DAC 1 */
seqData[9] = …;    /* DAC 2 */
seqData[12] = …;   /* DAC 5 */

/* 3rd Data set */
seqData[16] = …;   /* DAC 1 */
seqData[17] = …;   /* DAC 2 */
seqData[20] = …;   /* DAC 5 */

/* 4th Data set */
seqData[24] = …;   /* DAC 1 */
seqData[25] = …;   /* DAC 2 */
seqData[28] = …;   /* DAC 5 */

NumDataPages = 4;

/*----------------------------------------------------------------
  Write DAC data into driver’s FIFO
  ----------------------------------------------------------------*/
result = tpmc851DacSeqWrite(
              hdl,
              0,                           /* unused                   */
              &NumDataPages,
              seqData,                     /* Sequencer output data    */
              &seqStatus);
if (result == TPMC851_OK)
{
     /* function succeeded, check the number of transferred data pages */
     if (NumDataPages < 4) {
          /* remaining data has to be written again */
     }
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_NOMEM	No DAC sequencer FIFO available, sequencer not started
TPMC851_ERR_INVAL	Invalid flag, or buffer specified

3.3.10 tpmc851DacSeqStop

NAME

tpmc851DacSeqStop – Stop DAC Sequencer

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851DacSeqStop
(
        TPMC851_HANDLE        hdl
)
```

Class-method:

```
UInt32 DacSeqStop()
```

DESCRIPTION

This function stops the DAC sequencer. All sequencer channel configurations remain valid after stopping.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*------------------
  Stop DAC sequencer
  ------------------*/
result = tpmc851DacSeqStop( hdl );

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_ACCESS	The sequencer is not running

3.3.11 tpmc851IoRead

NAME

tpmc851IoRead – Read from digital I/O

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851IoRead
(
        TPMC851_HANDLE          hdl,
        unsigned short          *pIoValue
)
```

Class-method:

```
UInt32 IoRead
(
        ref UInt16              pIoValue
)
```

DESCRIPTION

This function reads the current value of digital I/O input.

PARAMETERS

hdl

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

pIoValue

This parameter specifies a pointer to an *unsigned short* value which receives the current I/O value. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE      hdl;
TPMC851_STATUS      result;
unsigned short      IoValue;

/*---------------------
  Read I/O input value
  ---------------------*/
result = tpmc851IoRead(
              hdl,
              &IoValue );                      /* I/O value   */

if (result == TPMC851_OK)
{
    /* function succeeded */
    printf("    I/O-value: 0x%04X", IoValue);
} else {
    /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.12 tpmc851IoWrite

NAME

tpmc851IoWrite – Write to digital I/O

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851IoWrite
(
        TPMC851_HANDLE          hdl,
        unsigned short          IoValue
)
```

Class-method:

```
UInt32 IoWrite
(
        UInt16                  IoValue
)
```

DESCRIPTION

This function writes a value to digital I/O output.

Only I/O lines configured for output will be affected. Please refer to chapter 3.3.13 tpmc851IoConfig.

PARAMETERS

hdl

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

IoValue

This value is written to the I/O output. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*------------------------------
  Write I/O output value 0x1234
  ------------------------------*/
result = tpmc851IoWrite(
              hdl,
              0x1234 );                        /* I/O value   */

if (result == TPMC851_OK)
{
    /* function succeeded */
} else {
    /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.13 tpmc851IoConfig

NAME

tpmc851IoConfig – Configure direction of digital I/O

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851IoConfig
(
        TPMC851_HANDLE        hdl,
        unsigned short        Direction
)
```

Class-method:

```
UInt32 IoConfig
(
        ref UInt16            Direction
)
```

DESCRIPTION

This function configures the direction (input/output) of the digital I/O lines.

PARAMETERS

hdl

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

Direction

Specifies the new direction setting for digital I/O. A bit set to 1 enables output, a 0 means that the I/O line is input. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*---------------------------------------------------------
  Enable line 0,2,8 for output, all other lines are input
  ---------------------------------------------------------*/
result = tpmc851IoConfig(
              hdl,
              (1 << 0) | (1 << 2) | (1 << 8) ); /* Direction */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.14 tpmc851IoDebConfig

NAME

tpmc851IoDebConfig – Configure digital I/O (input) debouncer

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851IoDebConfig
(
        TPMC851_HANDLE          hdl,
        unsigned short          EnableMask,
        unsigned short          DebounceTime
)
```

Class-method:

```
UInt32 IoDebConfig
(
        UInt16                  EnableMask,
        UInt16                  DebounceTime
)
```

DESCRIPTION

This function configures the digital I/O input debouncing mechanism to avoid detection of invalid signal changes in noisy environments.

PARAMETERS

hdl

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

EnableMask

Specifies digital I/O lines to be filtered by the debouncing mechanism. A bit set to 1 enables the debouncer, and a 0 disables the debouncer for the corresponding I/O line. Bit 0 corresponds to the first I/O line, bit 1 corresponds to the second I/O line and so on.

DebounceTime

Specifies the debounce time. The time is specified in 100ns steps, using the following formula:
Debounce duration = (DebounceTimeValue * 100ns) + 100ns

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-------------------------------------------------------
  Enable Debouncer for line 1 and 2 (debounce time 1ms)
  -------------------------------------------------------*/
result = tpmc851IoDebConfig(
              hdl,
              (1 << 1) | (1 << 2),    /* EnableMask                  */
              10000 );                /* DebounceTime (in 100ns steps) */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.15 tpmc851IoEventWait

NAME

tpmc851IoEventWait – Wait for I/O event

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851IoEventWait
(
        TPMC851_HANDLE          hdl,
        int                     IoLine,
        unsigned int            flags,
        int                     Timeout
)
```

Class-method:

```
UInt32 IoEventWait
(
        Int32                   IoLine,
        UInt32                  flags,
        Int32                   Timeout
)
```

DESCRIPTION

This function waits for an I/O input event.

PARAMETERS

hdl

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

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**
TPMC851_F_HI2LOTRANS	If set, the function will return after a high to low transition occurs.
TPMC851_F_LO2HITRANS	If set, the function will return after a low to high transition occurs.

At least one flag must be specified.

Timeout

Specifies the maximum time the function will wait for the specified event. The time shall be specified in milliseconds, but the timeout granularity is 1 second.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*---------------------------------------------------------
  Wait for any transition on I/O line 12 (max wait 10 sec)
  ---------------------------------------------------------*/
result = tpmc851IoEventWait(
              hdl,
              12,                                        /* IoLine     */
              TPMC851_F_HI2LOTRANS | TPMC851_F_LO2HITRANS,    /* Flags  */
              10000 );                                   /* Timeout    */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified
TPMC851_ERR_NOMEM	No free event object available
TPMC851_ERR_ACCESS	Invalid I/O line specified
TPMC851_ERR_TIMEOUT	Timeout has occurred
TPMC851_ERR_ABORTED	The waiting has been aborted by a power down of the system

3.3.16 tpmc851CntRead

NAME

tpmc851CntRead – Read counter/timer value

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851CntRead
(
        TPMC851_HANDLE          hdl,
        unsigned int            *pCounterValue,
        unsigned int            *pCounterStatus
)
```

Class-method:

```
UInt32 CntRead
(
        ref UInt32              pCounterValue,
        ref UInt32              pCounterStatus
)
```

DESCRIPTION

This function reads the value of the counter.

PARAMETERS

hdl

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

pCounterValue

This parameter is a pointer to an *unsigned int* data buffer where the current counter value is stored.

pCounterStatus

This parameter is a pointer to an *unsigned int* data buffer where the counter status is returned. If possible the flags are cleared after read. This is an ORed value of the following flags.

Flag	Description
TPMC851_SF_CNTBORROW	Counter borrow bit set (current state)
TPMC851_SF_CNTCARRY	Counter carry bit set (current state)
TPMC851_SF_CNTMATCH	Counter match event has occurred since last read.
TPMC851_SF_CNTSIGN	Counter sign bit (current state)
TPMC851_SF_CNTDIRECTION	If set, counter direction is upward. If not set, counter direction is downward.
TPMC851_SF_CNTLATCH	Counter value has been latched.
TPMC851_SF_CNTLATCHOVERFLOW	Counter latch overflow has occurred.
TPMC851_SF_CNTSNGLCYC	Counter Single Cycle is active

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE      hdl;
TPMC851_STATUS      result;
unsigned int        CounterValue;
unsigned int        CounterStatus;


/*------------------
  Read counter value
  ------------------*/
result = tpmc851CntRead(
              hdl,
              &CounterValue,
              &CounterStatus );

if (result == TPMC851_OK)
{
    /* function succeeded */
    printf("    Counter: %d", CounterValue);
    printf("    State:   %Xh", CounterStatus);
} else {
    /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.17 tpmc851CntConfig

NAME

tpmc851CntConfig – Configure counter

SYNOPSIS

<u>API-function:</u>

```
TPMC851_STATUS tpmc851CntConfig
(
          TPMC851_HANDLE        hdl,
          unsigned int          inputMode,
          int                   clockDivider,
          unsigned int          countMode,
          unsigned int          controlMode,
          unsigned int          invFlags
)
```

<u>Class-method:</u>

```
UInt32 CntConfig
(
          UInt32                inputMode,
          Int32                 clockDivider,
          UInt32                countMode,
          UInt32                controlMode,
          UInt32                invFlags
)
```

DESCRIPTION

This function configures the counter.

PARAMETERS

hdl

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

inputMode

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

Flag	Description
TPMC851_M_CNTIN_DISABLE	Counter disabled
TPMC851_M_CNTIN_TIMERUP	Timer Mode Up
TPMC851_M_CNTIN_TIMERDOWN	Timer Mode Down
TPMC851_M_CNTIN_DIRCOUNT	Direction Count
TPMC851_M_CNTIN_UPDOWNCOUNT	Up/Down Count
TPMC851_M_CNTIN_QUAD1X	Quadrature Count 1x
TPMC851_M_CNTIN_QUAD2X	Quadrature Count 2x
TPMC851_M_CNTIN_QUAD4X	Quadrature Count 4x

clockDivider

Specifies clock divider for Timer Mode. Allowed clock divider values are:

Clock Divider Value	Clock Input Frequency
1	40 MHz
2	20 MHz
4	10 MHz
8	5 MHz

countMode

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

Flag	Description
TPMC851_M_CNT_CYCLE	Cycling Counter
TPMC851_M_CNT_DIVN	Divide-by-N
TPMC851_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
TPMC851_M_CNTCTRL_NONE	No Control Mode
TPMC851_M_CNTCTRL_LOAD	Load Mode
TPMC851_M_CNTCTRL_LATCH	Latch Mode
TPMC851_M_CNTCTRL_GATE	Gate Mode
TPMC851_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**
TPMC851_F_CNTINVINP2	If set, input line 2 is low active If not set, input line 2 is high active
TPMC851_F_CNTINVINP3	If set, input line 3 is low active If not set, input line 3 is high active
TPMC851_F_CNTINVINP4	If set, input line 4 is low active If not set, input line 4 is high active

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*--------------------------------------------------------
  Setup counter for direction count, clock divider 1,
  cycling count, no control mode and all lines high active
  --------------------------------------------------------*/
result = tpmc851CntConfig(
              hdl,
              TPMC851_M_CNTIN_DIRCOUNT,    /* inputMode      */
              1,                           /* clockDivider   */
              TPMC851_M_CNT_CYCLE,         /* countMode      */
              TPMC851_M_CNTCTRL_NONE,      /* controlMode    */
              0 );                         /* invFlags       */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid mode or flag specified

3.3.18 tpmc851CntReset

NAME

tpmc851CntReset – Reset counter

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851CntReset
(
        TPMC851_HANDLE        hdl
)
```

Class-method:

UInt32 CntReset()

DESCRIPTION

This function resets the counter value to 0x00000000.

PARAMETERS

hdl

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

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

…

```
/*--------------------
  Reset counter value
  --------------------*/
result = tpmc851CntReset( hdl );

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.19 tpmc851CntSetPreload

NAME

tpmc851CntSetPreload – Set counter preload value

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851CntSetPreload
(
        TPMC851_HANDLE        hdl,
        unsigned int          PreloadValue,
        unsigned int          PreloadFlags
)
```

Class-method:

```
UInt32 CntSetPreload
(
        UInt32                PreloadValue,
        UInt32                PreloadFlags
)
```

DESCRIPTION

This function sets the counter preload value, either immediately or on the next preload condition.

PARAMETERS

hdl

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

PreloadValue

Specifies the new counter preload value.

PreloadFlags

The following flag is optional:

Flag	Description
TPMC851_F_IMMPRELOAD	If set, the function will immediately load the preload value into the counter If not set, preload value will be used for the next preload condition.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;


/*------------------------------------------
  Immediately load 0x11223344 into the counter
  and preload register
  ------------------------------------------*/
result = tpmc851CntSetPreload(
              hdl,
              0x11223344,                      /* Preload Value   */
              TPMC851_F_IMMPRELOAD );          /* Flags           */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_INVAL	Invalid flag specified

3.3.20 tpmc851CntSetMatch

NAME

tpmc851CntSetMatch – Set counter match value

SYNOPSIS

API-function:

```
TPMC851_STATUS tpmc851CntSetMatch
(
        TPMC851_HANDLE        hdl,
        unsigned int          MatchValue
)
```

Class-method:

```
UInt32 CntSetMatch
(
        UInt32                MatchValue
)
```

DESCRIPTION

This function sets the counter match value. If counter and match value are the same, a match event occurs. The driver can wait for this event.

PARAMETERS

hdl

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

MatchValue

Specifies the new counter match value.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;


/*--------------------------
  Set match value to 0x10000
  --------------------------*/
result = tpmc851CntSetMatch(
              hdl,
              0x10000 );              /* MatchValue */

if (result == TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid

3.3.21 tpmc851CntMatchWait

NAME

tpmc851CntMatchWait – Wait for counter match event

SYNOPSIS

<u>API-function:</u>

```
TPMC851_STATUS tpmc851CntMatchWait
(
        TPMC851_HANDLE        hdl,
        int                   timeout
)
```

<u>Class-method:</u>

```
UInt32 CntSetMatchWait
(
        Int32                 Timeout
)
```

DESCRIPTION

This function waits for a counter match event.

PARAMETERS

hdl

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

timeout

Specifies the maximum time the function will wait for the counter match event. The time shall be specified in milliseconds, but the timeout granularity is 1 second.

EXAMPLE

```
#include "tpmc851api.h"

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*----------------------------------------------------
  Wait for counter match event (max wait 10 sec)
  ----------------------------------------------------*/
result = tpmc851CntMatchWait( 10000 );

if (result != TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_NOMEM	No free event object available
TPMC851_ERR_TIMEOUT	Timeout has occurred
TPMC851_ERR_ABORTED	The waiting has been aborted by a power down of the system

3.3.22 tpmc851CntCtrlWait

NAME

tpmc851CntCtrlWait – Wait for counter control event

SYNOPSIS

<u>API-function:</u>

```
TPMC851_STATUS tpmc851CntCtrlWait
(
        TPMC851_HANDLE          hdl,
        int                     timeout
)
```

<u>Class-method:</u>

```
UInt32 CntCtrlWait
(
        Int32                   Timeout
)
```

DESCRIPTION

This function waits for counter control event. The event to wait for is chosen with API function *tpmc851CntConfig(),* specifying the parameter *controlMode*.

PARAMETERS

hdl

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

timeout

Specifies the maximum time the function will wait for the counter control event. The time shall be specified in milliseconds, but the timeout granularity is 1 second.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

/*-----------------------------------------------------
  Wait for counter control event (max wait 10 sec)
  -----------------------------------------------------*/
result = tpmc851CntCtrlWait( 10000 );

if (result != TPMC851_OK)
{
     /* function succeeded */
} else {
     /* handle error */
}
```

RETURN VALUE

On success, TPMC851_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid
TPMC851_ERR_NOMEM	No free event object available
TPMC851_ERR_TIMEOUT	Timeout has occurred
TPMC851_ERR_ABORTED	The waiting has been aborted by a power down of the system

4 Hibernate Mode

The driver supports the “Hibernate”-mode of Windows. The driver will remember the device configuration and the current output value and it will restore the configuration immediately after leaving the hibernation mode.

If there are pending requests, (e.g. waiting for an input transition, counter match or running ADC/DAC sequencer) and the system wants to enter hibernation mode, the request will be aborted with an appropriate error code. The application may now decide how to handle this situation.