The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC851-SW-82

Linux Device Driver

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

Version 2.0.x

User Manual

Issue 2.0.0

April 2024

TEWS Technologies GmbH
Eggerstedter Weg 14, 25421 Pinneberg, Germany
Phone: +49 (0) 4101 4058 0
e-mail: info@tews.com www.tews.com

TPMC851-SW-82

Linux Device Driver

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

Supported Modules:
TPMC851

Issue	Description	Date
1.0.0	First Issue	December 12, 2005
1.0.1	New Address TEWS LLC, ChangeLog.txt added to file list	December 3, 2006
1.0.2	General revision	November 26, 2008
1.0.3	Address of TEWS LLC removed	August 5, 2009
1.0.4	General revision	February 14, 2012
1.1.0	Correction of Counter Input Mode definition (TPMC851_M_CNTIN_QUAD3X to TPMC851_M_CNTIN_QUAD4X) Correction of value for wait without timeout	December 3, 2014
1.1.1	File-List modified	September 24, 2019
1.2.0	Additional Function TPMC851_IOC_ADC_SEQWAITREAD implemented	July 19, 2023
1.2.1	Correction of Example TPMC851_IOC_ADC_SEQWAITREAD	July 27, 2023
2.0.0	API Interface implemented Additional Counter-Channels for newer board revisions New Address of TEWS Technologies GmbH	April 19, 2024

Table of Contents

1 Introduction

The TPMC851-SW-82 Linux device driver allows the operation of a TPMC851 PMC on Linux operating systems.

Supported features:

- Executing AD conversion and reading input value
- Setting up, Starting and Stopping ADC Input Sequencer
- Configuring ADC Sequencer Trigger I/O
- Reading ADC Sequencer input data (immediate or waiting)
- Setting output value and starting DA conversion
- Setting up, Starting and Stopping DAC Sequencer
- Configuring DAC Sequencer Trigger I/O
- Setting DAC Sequencer Data
- Reading digital I/O data
- Setting digital output data
- Configuring I/O direction and input debouncer
- Waiting for input events
- Reading counter value
- Resetting counter value
- Configuring counter mode and controls
- Setting preload and match value
- Waiting for counter events

The TPMC851-SW-82 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 usage of TPMC851 devices it is recommended to read the manuals listed below.

TPMC851 User Manual

2 Installation

The directory TPMC851-SW-82 on the distribution media contains the following files:

TPMC851-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
TPMC851-SW-82-2.0.0.pdf	PDF copy of this manual
Release.txt	Release information
ChangeLog.txt	Release history

The GZIP compressed archive TPMC851-SW-82-SRC.tar.gz contains the following files and directories:

Directory path './tpmc851/':

tpmc851.c	TPMC851 device driver source
tpmc851def.h	TPMC851 driver include file
tpmc851.h	TPMC851 include file for driver and application
makenode	Script to create device nodes on the file system
Makefile	
api/tpmc851api.c	API source file
api/tpmc851api.h	API include file
include/config.h	Driver independent library header file
include/tpmodule.h	Driver and kernel independent library header file
include/tpmodule.c	Driver and kernel independent library source file
include/tpxxxhwdep.h	HAL library header file
include/tpxxxhwdep.c	HAL library source file
example/tpmc851exa.c	Example application
example/Makefile	Example application make file
COPYING	Copy of the GNU Public License (GPL)

In order to perform an installation, extract all files of the archive TPMC851-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC851-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tpmc851.h and tpmc851api.h to */usr/include*

2.1 Build and install the Device Driver

- Login as *root*
- Change to the target directory
- To create and install the driver in the module directory */lib/modules/<version>/misc* enter:

  # make install

- Only after the first build we have to execute *depmod* to create a new dependency description for loadable kernel modules. This dependency file is later used by *modprobe* to automatically load dependent kernel modules.

  # depmod –aq

2.2 Uninstall the Device Driver

- Login as *root*
- Change to the target directory
- To remove the driver from the module directory */lib/modules/<version>/misc* enter:

  # make uninstall

- Update kernel module dependency description file

  # depmod –aq

2.3 Install Device Driver into the running Kernel

- To load the device driver into the running kernel, login as root and execute the following commands:

  # modprobe tpmc851drv

- After the first build or if you are using dynamic major device allocation it is necessary to create new device nodes on the file system. Please execute the script file *makenode* to do this. If your kernel has enabled a dynamic device file system (devfs or sysfs with udev) then you have to skip running the *makenode* script. Instead of creating device nodes from the script the driver itself takes creating and destroying of device nodes in its responsibility.

  # sh makenode

On success the device driver will create a minor device for each compatible channel found. The first PMC module can be accessed with device node /dev/tpmc851_0, the second module with device node /dev/tpmc851_1 and so on.

The assignment of device nodes to physical PMC modules depends on the search order of the PCI bus driver.

2.4 Remove Device Driver from the running Kernel

- To remove the device driver from the running kernel login as root and execute the following command:

  # modprobe –r tpmc851drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc851_* nodes will be automatically removed from your file system after this.

Be sure that the driver isn't opened by any application program. If opened you will get the response "*tpmc851drv: Device or resource busy*" and the driver will still remain in the system until you close all opened files and execute *modprobe –r* again.

2.5 Change Major Device Number

The TPMC851 driver uses dynamic allocation of major device numbers by default. If this isn't suitable for the application it is possible to define a major number for the driver. If the kernel has enabled devfs the driver will not use the symbol TPMC851_MAJOR.

To change the major number edit the file *tpmc851def.h*, change the following symbol to appropriate value and enter **make install** to create a new driver.

TPMC851_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC851_MAJOR   122
```

Be sure that the desired major number isn't used by other drivers. Please check /proc/devices to see which numbers are free.

3 API Documentation

3.1 General Functions

3.1.1 tpmc851Open

NAME

tpmc851Open – Opens a Device

SYNOPSIS

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

DESCRIPTION

Before I/O can be performed to a device, a file descriptor must be opened by a call to this function.

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tpmc851Open(“/dev/tpmc851_0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. An error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno.*

The error code is a standard error code set by the I/O system.

3.1.2 tpmc851Close

NAME

tpmc851Close – Closes a Device

SYNOPSIS

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

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

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

EXAMPLE

```
#include “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

TPMC851_ERR_INVALID_HANDLE	The specified TPMC851_HANDLE is invalid.

Other returned error codes are system error conditions.

3.1.3 tpmc851GetPciInfo

NAME

tpmc851GetPciInfo – get information of the module PCI header

SYNOPSIS

```
TPMC851_STATUS tpmc851GetPciInfo
(
    TPMC851_HANDLE          hdl,
    TPMC851_PCIINFO_BUF     *pPciInfoBuf
)
```

DESCRIPTION

This function returns information of the module PCI header in the provided data buffer.

PARAMETERS

hdl

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

pPciInfoBuf

This argument is a pointer to the structure TPMC851_PCIINFO_BUF that receives information of the module PCI header.

```
typedef struct
{
    unsigned short      vendorId;
    unsigned short      deviceId;
    unsigned short      subSystemId;
    unsigned short      subSystemVendorId;
    int                 pciBusNo;
    int                 pciDevNo;
    int                 pciFuncNo;
} TPMC851_PCIINFO_BUF;
```

vendorId

PCI module vendor ID.

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module sub system vendor ID

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE          hdl;
TPMC851_STATUS          result;
TPMC851_PCIINFO_BUF     pciInfoBuf

/*
** get module PCI information
*/
result = tpmc851GetPciInfo( hdl, &pciInfoBuf );

if (result != TPMC851_OK)
{
     /* 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 device handle is invalid

3.1.4 tpmc851GetBoardInfo

NAME

tpmc851GetBoardInfo – get hardware information from the module

SYNOPSIS

```
TPMC851_STATUS tpmc851GetBoardInfo
(
    TPMC851_HANDLE          hdl,
    TPMC851_BOARDINFO_BUF   *pBoardInfoBuf
)
```

DESCRIPTION

This function returns information about the module, e.g. firmware id (version) and the number of available counter channels.

PARAMETERS

hdl

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

pBoardInfoBuf

This argument is a pointer to the structure TPMC851_BOARDINFO_BUF that receives information about the hardware.

```
typedef struct
{
    unsigned int    firmwareId;
    int             numCounterChannels;
    int             numAdcChannels;
    int             numDacChannels;
    int             numDioChannels;
    unsigned int    adcVoltageRange;
} TPMC851_BOARDINFO_BUF;
```

firmwareId

Firmware Version ID (refer to hardware manual).

numCounterChannels

Number of Counter channels supported by the hardware.

numAdcChannels

Number of ADC channels supported by the hardware.

numDacChannels

Number of DAC channels supported by the hardware.

numDioChannels

Number of Digital I/O channels supported by the hardware.

adcVoltageRange

Full-Scale ADC input voltage, in millivolts. Can be used to calculate the actual voltage out of the digital ADC value. For TPMC851-10R, this value is 10000.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE          hdl;
TPMC851_STATUS          result;
TPMC851_BOARDINFO_BUF   BoardInfo;

/*
** get module board information
*/
result = tpmc851GetBoardInfo( hdl, &BoardInfo );
if (result != TPMC851_OK)
{
     /* handle error */
}

printf(“Firmware-ID: 0x%08X\n”, BoardInfo.firmwareId);
```

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 device handle is invalid

3.1.5 tpmc851GetHwInfo

NAME

tpmc851GetHwInfo – get hardware information

SYNOPSIS

```
TPMC851_STATUS tpmc851GetHwInfo
(
    TPMC851_HANDLE      hdl,
    unsigned int        hwInfoType,
    unsigned int        *hwInfoValue
)
```

DESCRIPTION

This function returns the requested hardware information in the provided data buffer.

PARAMETERS

hdl

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

hwInfoType

This argument specifies the requested hardware value type. The following values are valid:

Value	Description
TPMC851_HWINFO_PCIVENDORID	PCI Vendor ID (16 bit)
TPMC851_HWINFO_PCIDEVICEID	PCI Device ID (16 bit)
TPMC851_HWINFO_PCISUBSYSID	PCI Subsystem ID (16 bit)
TPMC851_HWINFO_PCISUBSYSVENDID	PCI Subsystem Vendor ID (16 bit)
TPMC851_HWINFO_PCIBUSNO	PCI Bus Number
TPMC851_HWINFO_PCIDEVNO	PCI Device Number
TPMC851_HWINFO_PCIFUNCNO	PCI Function Number
TPMC851_HWINFO_FIRMWAREID	Firmware ID
TPMC851_HWINFO_CNTCHANS	Number of supported Counter Channels
TPMC851_HWINFO_ADCCHANS	Number of supported ADC Channels
… continued	

Value	Description
TPMC851_HWINFO_DACCHANS	Number of supported DAC Channels
TPMC851_HWINFO_DIOCHANS	Number of supported Digital I/O Channels
TPMC851_HWINFO_ADCVOLTRANGE	Full-Scale ADC input voltage, in millivolts. Can be used to calculate the actual voltage out of the digital ADC value. For TPMC851-10R, this value is 10000.

hwInfoValue

This argument returns the requested hardware information value.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;
unsigned int       busNo, devNo;

/*
** get module PCI localization
*/
result = tpmc851GetHwValue( hdl, TPMC851_HWVAL_PCIBUSNO, &busNo );
if (result != TPMC851_OK)
{
     /* handle error */
}
result = tpmc851GetHwValue( hdl, TPMC851_HWVAL_PCIDEVNO, &devNo );
/* handle return/error code ... */

printf(“PCI Localization: bus=%d dev=%d\n”, busNo, devNo);
```

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 device handle is invalid

3.2 Device Access Interface

3.2.1 tpmc851AdcRead

NAME

tpmc851AdcRead – Read value from ADC channel

SYNOPSIS

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

3.2.2 tpmc851AdcSeqConfig

NAME

tpmc851AdcSeqConfig – Configure ADC sequencer channel

SYNOPSIS

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

3.2.3 tpmc851AdcSeqStart

NAME

tpmc851AdcSeqStart – Start ADC Sequencer

SYNOPSIS

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

3.2.4 tpmc851AdcSeqRead

NAME

tpmc851AdcSeqRead – Read ADC Sequencer Data

SYNOPSIS

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

This parameter is not supported

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,
              0,
              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

3.2.5 tpmc851AdcSeqReadWait

NAME

tpmc851AdcSeqReadWait – Read ADC Sequencer Data and wait for new data

SYNOPSIS

```
TPMC851_STATUS tpmc851AdcSeqReadWait
(
    TPMC851_HANDLE      hdl,
    unsigned int        flags,
    int                 *pData,
    unsigned int        timeout,
    unsigned int        *pStatus
)
```

DESCRIPTION

This function reads data of one sequencer cycle from the ADC sequencer buffer. The function returns if data is received, or a specified time has expired.

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 not supported

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.

timeout

Specifies the maximum time the function will wait for new ADC data. The time is specified in milliseconds.

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 or wait the specified time (5 seconds)
  -----------------------------------------------------*/
result = tpmc851AdcSeqReadWait(
              hdl,
              0,
              adcSeqBuf,
              5000,
              &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_TIMEOUT	The read function (waiting for data) exceeds the specified time
TPMC851_ERR_INVAL	Invalid flag, or buffer specified

3.2.6 tpmc851AdcSeqStop

NAME

tpmc851AdcSeqStop – Stop ADC Sequencer

SYNOPSIS

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

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

3.2.7 tpmc851DacWrite

NAME

tpmc851DacWrite – Write to DAC channel

SYNOPSIS

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

3.2.8 tpmc851DacSeqConfig

NAME

tpmc851DacSeqConfig – Configure DAC sequencer channel

SYNOPSIS

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

3.2.9 tpmc851DacSeqStart

NAME

tpmc851DacSeqStart – Start DAC Sequencer

SYNOPSIS

```
TPMC851_STATUS tpmc851DacSeqStart
(
    TPMC851_HANDLE      hdl,
    unsigned short      cycTime,
    unsigned int        flags,
    unsigned int        NumOfBufferPages,
    unsigned int        NumDataPages,
    short               *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 a short value for every enabled 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 1st enabled channel at index 0, the date for the 2md channel at index 1, and so on. The data for the 2nd cycle follows directly behind the data for the enable channel of the 1st cycle. If there are 4 active channels the data for the 2nd cycle starts at index 4 with the first enabled channel and so on.

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 5
1st	0	1	2
2nd	3	4	5
3rd	6	7	8
4th	9	10	11

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE              hdl;
TPMC851_STATUS              result;
int                         seqData[3*4];      /* (3 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[2] = …;    /* DAC 5 */

/* 2nd Data set */
seqData[3] = …;    /* DAC 1 */
seqData[4] = …;    /* DAC 2 */
seqData[5] = …;    /* DAC 5 */

/* 3rd Data set */
seqData[6] = …;    /* DAC 1 */
seqData[7] = …;    /* DAC 2 */
seqData[8] = …;    /* DAC 5 */

/* 4th Data set */
seqData[9] = …;    /* DAC 1 */
seqData[10] = …;   /* DAC 2 */
seqData[11] = …;   /* 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

3.2.10 tpmc851DacSeqWrite

NAME

tpmc851DacSeqWrite – Transfer DAC Sequencer data into the sequencer buffer

SYNOPSIS

```
TPMC851_STATUS tpmc851DacSeqWrite
(
    TPMC851_HANDLE      hdl,
    unsigned int        flags,
    unsigned int        *NumDataPages,
    short               *pData,
    unsigned int        *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 a short value for every enabled 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 1st enabled channel at index 0, the date for the 2md channel at index 1, and so on. The data for the 2nd cycle follows directly behind the data for the enable channel of the 1st cycle. If there are 4 active channels the data for the 2nd cycle starts at index 4 with the first enabled channel and so on.

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 5
1st	0	1	2
2nd	3	4	5
3rd	6	7	8
4th	9	10	11

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[3*4];      /* (3 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[2] = …;    /* DAC 5 */

/* 2nd Data set */
seqData[3] = …;    /* DAC 1 */
seqData[4] = …;    /* DAC 2 */
seqData[5] = …;    /* DAC 5 */

/* 3rd Data set */
seqData[6] = …;    /* DAC 1 */
seqData[7] = …;    /* DAC 2 */
seqData[8] = …;    /* DAC 5 */

/* 4th Data set */
seqData[9] = …;    /* DAC 1 */
seqData[10] = …;   /* DAC 2 */
seqData[11] = …;   /* 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

3.2.11 tpmc851DacSeqStop

NAME

tpmc851DacSeqStop – Stop DAC Sequencer

SYNOPSIS

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

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

3.2.12 tpmc851IoRead

NAME

tpmc851IoRead – Read from digital I/O

SYNOPSIS

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

3.2.13 tpmc851IoWrite

NAME

tpmc851IoWrite – Write to digital I/O

SYNOPSIS

```
TPMC851_STATUS tpmc851IoWrite
(
    TPMC851_HANDLE      hdl,
    unsigned short      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 3.2.14 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

3.2.14 tpmc851IoConfig

NAME

tpmc851IoConfig – Configure direction of digital I/O

SYNOPSIS

```
TPMC851_STATUS tpmc851IoConfig
(
      TPMC851_HANDLE      hdl,
      unsigned short      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

3.2.15 tpmc851IoDebConfig

NAME

tpmc851IoDebConfig – Configure digital I/O (input) debouncer

SYNOPSIS

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

3.2.16 tpmc851IoEventWait

NAME

tpmc851IoEventWait – Wait for I/O event

SYNOPSIS

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

3.2.17 tpmc851CntRead

NAME

tpmc851CntRead – Read counter/timer value of specified channel

SYNOPSIS

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

DESCRIPTION

This function reads the value of the specified counter channel.

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

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 of channel 1
  ------------------*/
result = tpmc851CntRead(
              hdl,
              1,                              /* Channel Number  */
              &CounterValue,                  /* Counter Value   */
              &CounterStatus );               /* Counter Status  */

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
TPMC851_ERR_ACCESS	Invalid counter channel specified

3.2.18 tpmc851CntConfig

NAME

tpmc851CntConfig – Configure specified counter channel

SYNOPSIS

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

DESCRIPTION

This function configures the specified counter channel.

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

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 channel 1 for direction count, clock divider 1,
  cycling count, no control mode and all lines high active
  -------------------------------------------------------*/
result = tpmc851CntConfig(
              hdl,
              1,                          /* Channel Number  */
              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
TPMC851_ERR_ACCESS	Invalid counter channel specified

3.2.19 tpmc851CntReset

NAME

tpmc851CntReset – Reset specified counter channel

SYNOPSIS

```
TPMC851_STATUS tpmc851CntReset
(
    TPMC851_HANDLE      hdl,
    int                 channelNo
)
```

DESCRIPTION

This function resets the counter value of the specified channel to 0x00000000.

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

…

```
/*-------------------
  Reset value of counter channel 1
  -------------------*/
result = tpmc851CntReset( hdl, 1 );
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	Invalid counter channel specified

3.2.20 tpmc851CntSetPreload

NAME

tpmc851CntSetPreload – Set counter preload value of specified channel

SYNOPSIS

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

DESCRIPTION

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

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

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 counter channel 1
  and preload register
  -----------------------------------------*/
result = tpmc851CntSetPreload(
              hdl,
              1,                              /* Channel number  */
              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
TPMC851_ERR_ACCESS	Invalid counter channel specified

3.2.21 tpmc851CntSetMatch

NAME

tpmc851CntSetMatch – Set counter match value of specified channel

SYNOPSIS

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

DESCRIPTION

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

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

MatchValue

Specifies the new counter match value.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*--------------------------
  Set match value of channel 1 to 0x10000
  --------------------------*/
result = tpmc851CntSetMatch(
              hdl,
              1,                      /* Channel number  */
              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
TPMC851_ERR_ACCESS	Invalid counter channel specified

3.2.22 tpmc851CntMatchWait

NAME

tpmc851CntMatchWait – Wait for counter match event on specified channel

SYNOPSIS

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

DESCRIPTION

This function waits for a counter match event on the specified channel.

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

timeout

Specifies the maximum time the function will wait for the counter match event. The time is specified in milliseconds. Specify -1 to wait indefinitely.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE     hdl;
TPMC851_STATUS     result;

…
```

```
…

/*-----------------------------------------------
  Wait for counter match event on channel 1 (max wait 10000 milliseconds)
  -----------------------------------------------*/
result = tpmc851CntMatchWait(
              hdl,
              1,                          /* Channel number  */
              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_NOMEM	No free event object available
TPMC851_ERR_TIMEOUT	Timeout has occurred
TPMC851_ERR_ACCESS	Invalid counter channel specified

3.2.23 tpmc851CntCtrlWait

NAME

tpmc851CntCtrlWait – Wait for counter control event on specified channel

SYNOPSIS

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

DESCRIPTION

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

PARAMETERS

hdl

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

channelNo

This parameter specifies the desired counter channel. Valid values are 1 up to the available number of channels (depends on hardware version).

timeout

Specifies the maximum time the function will wait for the counter control event. The time is specified in milliseconds. Specify -1 to wait indefinitely.

EXAMPLE

```
#include “tpmc851api.h”

TPMC851_HANDLE      hdl;
TPMC851_STATUS      result;

…
```

```
…

/*-----------------------------------------------------
  Wait for counter control event (max wait 10000 milliseconds)
     at counter channel 1.
  -----------------------------------------------------*/
result = tpmc851CntCtrlWait(
              hdl,
              1,                          /* Channel number  */
              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_NOMEM	No free event object available
TPMC851_ERR_TIMEOUT	Timeout has occurred
TPMC851_ERR_ACCESS	Invalid counter channel specified

4 Diagnostic

If the TPMC851 does not work properly it is helpful to get some status information from the driver respective kernel.

Check TPMC851 PCI information with lspci, which displays the PCI location of the TPMC851 and its addresses.

```
lspci –v
…
05:00.0 Signal processing controller: TEWS Technologies GmbH Device 0353
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: slow devsel, IRQ 18
        Memory at a1103000 (32-bit, non-prefetchable) [size=128]
        I/O ports at 4000 [size=128]
        Memory at a1102000 (32-bit, non-prefetchable) [size=512]
        Memory at a1101000 (32-bit, non-prefetchable) [size=64]
        Memory at a1100000 (32-bit, non-prefetchable) [size=64]
        Kernel driver in use: TEWS Technologies TPMC851 AD-DA-Converter,
Digital IO and Counter
        Kernel modules: tpmc851drv
…
```

The Linux */proc* file system provides information about kernel, resources, driver, devices and so on. The following screen dumps displays information of a correct running TPMC851 driver (see also the proc man pages).

```
cat /proc/devices
Character devices:
  1 mem
  2 tty
…
236 tpmc851drv
…

# cat /proc/iomem
…
8f800000-dfffffff : PCI Bus 0000:00
  a1100000-a11fffff : PCI Bus 0000:04
    a1100000-a11fffff : PCI Bus 0000:05
      a1100000-a110003f : 0000:05:00.0
        a1100000-a110003f : TPMC851
      a1101000-a110103f : 0000:05:00.0
        a1101000-a110103f : TPMC851
      a1102000-a11021ff : 0000:05:00.0
        a1102000-a11021ff : TPMC851
      a1103000-a110307f : 0000:05:00.0
…
```