The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC151-SW-42

VxWorks Device Driver

4 Channel Resolver or LVDT/RVDT-to-Digital Converter

Version 1.0.x

User Manual

Issue 1.0.0

January 2025

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

TPMC151-SW-42

VxWorks Device Driver

4 Channel Resolver or
LVDT/RVDT-to-Digital Converter

Supported Modules:
TPMC151

This document contains information, which is proprietary to TEWS Technologies GmbH. Any reproduction without written permission is forbidden.

TEWS Technologies GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS Technologies GmbH reserves the right to change the product described in this document at any time without notice.

TEWS Technologies GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2025 by TEWS Technologies GmbH

Issue	Description	Date
1.0.0	First Issue	January 29, 2025

Table of Contents

1 Introduction

The TPMC151-SW-42 VxWorks device driver software allows the operation of the supported PMC conforming to the VxWorks I/O system specification.

The driver provides an application programming interface (API) which allows OS independent access to the devices for compatibility between different OS versions and OS.

The TPMC151-SW-42 release contains independent driver sources for VxBus-enabled (GEN1 and GEN2) driver model. The VxBus-enabled driver is recommended for new developments with later VxWorks 6.x release and mandatory for VxWorks 64-bit and SMP systems.

The driver invokes a mutual exclusion and binary semaphore mechanism to prevent simultaneous requests by multiple tasks from interfering with each other.

The TPMC151-SW-42 device driver supports the following features:

- Reading angle and status from specified channels
- Reading angle and status from a pair of combined channels
- Reading a set (around a trigger) of angles from specified channel
- Configuration of Excitation of a specified channel
- General board configuration
- Support of on-board interval timer
- Wait for supported Events

The TPMC151-SW-42 supports the modules listed below:

TPMC151-10R	4 Channel Resolver or LVDT/RVDT-to-Digital Converter	(PMC)
TPMC151-20R	2 Channel Resolver or LVDT/RVDT-to-Digital Converter and 2 Channel Synchro-to-Digital Converter	(PMC)

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

TPMC151 User Manual
TEWS Technologies VxWorks Device Drivers - Installation Guide

2 API Documentation

2.1 General Functions

2.1.1 tpmc151Open

NAME

tpmc151Open – opens a device.

SYNOPSIS

```
TPMC151_HANDLE tpmc151Open
(
      char          *devicename
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle 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. The first TPMC151 device is named “/tpmc151/0”, the second device is named “/tpmc151/1” and so on.

EXAMPLE

```
#include “tpmc151api.h”

TPMC150_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tpmc151Open(“/tpmc151/0”);
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.

2.1.2 tpmc151Close

NAME

tpmc151Close – closes a device.

SYNOPSIS

```
TPMC151_STATUS tpmc151Close
(
     TPMC151_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 “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*
** close the device
*/
result = tpmc151Close(hdl);
if (result != TPMC151_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.3 tpmc151GetBoardInfo

NAME

tpmc151GetBoardInfo – get hardware information from the module

SYNOPSIS

```
TPMC151_STATUS tpmc151GetBoardInfo
(
    TPMC151_HANDLE              hdl,
    TPMC151_PCIINFO_BUF         *pPciInfoBuf,
    TPMC151_MODULEINFO_BUF      *pModuleInfoBuf
)
```

DESCRIPTION

This function returns information about the module, e.g. PCI location, firmware id (version), or available interfaces.

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 TPMC151_PCIINFO_BUF that receives information with the PCI identifiers and PCI location.

```
typedef struct
{
    unsigned short      vendorId;
    unsigned short      deviceId;
    unsigned short      subSystemId;
    unsigned short      subSystemVendorId;
    int                 pciBusNo;
    int                 pciDevNo;
    int                 pciFuncNo;
} TPMC151_PCIINFO_BUF;
```

vendorId

Returns the PCI vendor ID of the board.

deviceId

Returns the PCI device ID of the board.

subSystemId

Returns the PCI subsystem ID of the board.

subSystemVendord

Returns the PCI subsystem vendor ID of the board.

pciBusNo

Returns the PCI bus number.

pciDevNo

Returns the PCI device number

pciFuncNo

Returns the PCI function number.

pModuleInfoBuf

This argument is a pointer to the structure TPMC151_MODULEINFO_BUF that receives information about the hardware.

```
typedef struct
{
    char            moduleType[20];
    unsigned int    firmwareId;
    unsigned int    channelType[TPMC151_NUM_MAX_CHANS];
} TPMC151_MODULEINFO_BUF;
```

moduleType

Returns a null terminated string, containing the full module type, e.g. “TPMC151-10R”.

firmwareId

Returns the firmware version used on the board. The version is returned in 32 bit word in the following format:

```
MMmmRRbb        (hex-format)
MM              major version,
  mm            minor version,
    RR          revision
      bb        build version
```

e.g. Firmware ID 1.2.3.4 will be returned as 0x01020304

channelType

This array returns the interface configuration of the available channels. Index 0 returns the interface of channel 1, Index 1 returns the interface of channel 2, and so on.
The following interfaces are defined:

Value	Description
TPMC151_CHANTYPE_NONE	No interface available
TPMC151_CHANTYPE_RESOLVER	Resolver or LVDT/RVDT-to-Digital Converter Interface
TPMC151_CHANTYPE_SYNCHRO	Synchro-to-Digital Converter Interface

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE          hdl;
TPMC151_STATUS          result;
TPMC151_PCIINFO_BUF     pciInfo;
TPMC151_MODULEINFO_BUF  moduleInfo;

/*
** get module board information
*/
result = tpmc151GetBoardInfo( hdl, &pciInfo, &moduleInfo );
if (result != TPMC151_OK)
{
     /* handle error */
}

printf(“Board: %s\n”, molduleInfo.moduleType);
printf(“Firmware-ID: %02d.%02d.%02d Build: %02d.\n”,
     (moduleInfo.firmwareId >> 24) & 0xFF,
     (moduleInfo.firmwareId >> 16) & 0xFF,
     (moduleInfo.firmwareId >> 8) & 0xFF,
     moduleInfo.firmwareId & 0xFF);
printf(“PCI-Location: %d:%d:%d.\n”,
     pciInfo.pciBusNo, pciInfo.pciDevNo, pciInfo.pciFuncNo);
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.4 tpmc151GetBoardHealth

NAME

tpmc151GetBoardHealth – get hardware information from the module

SYNOPSIS

```
TPMC151_STATUS tpmc151GetBoardInfo
(
    TPMC151_HANDLE          hdl,
    int                     *pAdcTemperature
)
```

DESCRIPTION

This function returns information about the module health, e.g. the on-board temperature of the XADC.

PARAMETERS

hdl

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

pAdcTemperature

This argument is a pointer to an integer value returning the on-chip temperature of the XADC on the TPMC151. The returned temperature is scaled to $^{1}/_{1000}$ °C.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE          hdl;
TPMC151_STATUS          result;
int                     temperature;

…
```

…

```
/*
** get module health information
*/
result = tpmc151GetBoardHealth( hdl, &temperature );
if (result != TPMC151_OK)
{
     /* handle error */
}

printf("XADC-temperature: %8.4f °C\n", temperature / 1000.0);
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified device handle is invalid

2.2 Device Access Functions

2.2.1 tpmc151ReadValueStatus

NAME

tpmc151ReadValueStatus – Read angle / stroke value and the current status of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadValueStatus
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned short      *value,
    unsigned short      *status
)
```

DESCRIPTION

This function reads the current angle / stroke value and status from the specified channel.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

value

This argument is a pointer to an unsigned short integer value returning the current angle / stroke value read from the channel.

status

This argument is a pointer to an unsigned short integer value returning the current status of the channel. The status is an ORed value of the following flags:

Flag	Description
TPMC151_STAT_QUAD	Quadrant-Detection Error detected, data is not valid. Possible causes are: - 180° jump - Excessive rotation speed - Excessive offsets - Excessive phase shifts (> 45°)
TPMC151_STAT_LOS	SIN/COS Loss-of-Signal, SIN and COS signal is below 1/16 of the selected input range.
TPMC151_STAT_CLIP_SIN	SIN Clipping, data is not valid.
TPMC151_STAT_CLIP_COS	COS Clipping, data is not valid.
TPMC151_STAT_EXC_LOW	Excitation frequency is too low.
TPMC151_STAT_EXC_HIGH	Excitation frequency is too high.
TPMC151_STAT_LOF	RDC entered low frequency mode.
TPMC151_STAT_INIT_DONE	RDC initialization done & output data is valid.
TPMC151_STAT_AMP_OT	The excitation amplifier signaled an overtemperature condition.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned short     value;
unsigned short     status;
char               statusStr[200];

/*-------------------------------------------------------------
  Read the current angle / stroke value and status of channel 1
  -------------------------------------------------------------*/
result = tpmc151ReadValueStatus( hdl, 1, &value, &status );
if (result == TPMC151_OK)
{
     /* handle error */
}

…
```

…

```
/* function succeeded */
printf("Ang./Str. = %d – Status: %04X\n", value, status);
if (status!= 0)
{
     sprintf(statusStr, "    ");
     if (status & TPMC151_STAT_QUAD)
          sprintf(statusStr, "%s QUAD", statusStr);
     if (status & TPMC151_STAT_LOS)
          sprintf(statusStr, "%s LOS", statusStr);
     if (status & TPMC151_STAT_CLIP_SIN)
          sprintf(statusStr, "%s CLIP_SIN", statusStr);
     if (status & TPMC151_STAT_CLIP_COS)
          sprintf(statusStr, "%s CLIP_COS", statusStr);
     if (status & TPMC151_STAT_EXC_LOW)
          sprintf(statusStr, "%s EXC_LOW", statusStr);
     if (status & TPMC151_STAT_EXC_HIGH)
          sprintf(statusStr, "%s EXC_HIGH", statusStr);
     if (status & TPMC151_STAT_LOF)
          sprintf(statusStr, "%s LOF", statusStr);
     if (status & TPMC151_STAT_INIT_DONE)
          sprintf(statusStr, "%s INIT_DONE", statusStr);
     if (status & TPMC151_STAT_AMP_OT)
          sprintf(statusStr, "%s AMP_OT", statusStr);
     printf("%s\n", statusStr);
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.2 tpmc151ReadValueVelocity

NAME

tpmc151ReadValueVelocity – Read angle / stroke value and the velocity of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadValueVelocity
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned short      *value,
    short               *velocity,
    unsigned short      *status
)
```

DESCRIPTION

This function reads the current angle / stroke value and velocity from the specified channel.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

value

This argument is a pointer to an unsigned short integer value returning the current angle / stroke value read from the channel.

velocity

This argument is a pointer to a short integer value returning the current velocity from the channel.

status

This argument is a pointer to an unsigned short integer value returning the current status of the channel. The status is an ORed value of the following flags:

Flag	Description
TPMC151_VEL_QUAD	Quadrant-Detection Error detected, data is not valid. Possible causes are: - 180° jump - Excessive rotation speed - Excessive offsets - Excessive phase shifts (> 45°)
TPMC151_VEL_CLIP	SIN/COS Clipping, data is not valid.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned short     value;
short              velocity;
unsigned short     status;
char               statusStr[200];

/*-----------------------------------------------------------------------
  Read the current angle / stroke value, velocity and status of channel 1
  -----------------------------------------------------------------------*/
result = tpmc151ReadValueVelocity( hdl, 1, &value, &velocity, &status);
if (result == TPMC151_OK)
{
     /* handle error */
}

/* function succeeded */
printf("Ang./Str. = %d – Vel. = %d - Status: %04X\n", value, velocity,
status);
if (status!= 0)
{
     sprintf(statusStr, "    ");
     if (status & TPMC151_VEL_QUAD)
          sprintf(statusStr, "%s QUAD", statusStr);
     if (status & TPMC151_VEL_CLIP)
          sprintf(statusStr, "%s CLIP ", statusStr);
     printf("%s\n", statusStr);
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.3 tpmc151ReadValueIndex

NAME

tpmc151ReadValueIndex – Read angle / stroke value and a sample count index of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadValueIndex
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned short      *value,
    unsigned short      *index
)
```

DESCRIPTION

This function reads the current angle / stroke value and a sample count from the specified channel.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

value

This argument is a pointer to an unsigned short integer value returning the current angle / stroke value read from the channel.

index

This argument is a pointer to an unsigned short integer value returning the sample count index from the channel.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned short     value;
unsigned short     index;

/*-------------------------------------------
  Read the current angle and index of channel 1
  -------------------------------------------*/
result = tpmc151ReadValueIndex( hdl, 1, &value, &index );
if (result == TPMC151_OK)
{
     /* handle error */
}

/* function succeeded */
printf("Ang./Str. = %d – Index = %d\n", angle, index);
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.4 tpmc151ReadCombValueStatus

NAME

tpmc151ReadCombValueStatus – Read angle / stroke value and the current status synchronous from specified channel pair

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadCombValueStatus
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned short      *valueA,
    unsigned short      *valueB,
    unsigned short      *statusA
    unsigned short      *statusB
)
```

DESCRIPTION

This function reads the current angle and status synchronous from the specified channel pair.

PARAMETERS

hdl

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

channel

Specifies the channel of the first channel of the channel pair. Valid values are 1 to read channel pair 1/2 and 3 to read channel pair 3/4

valueA

This argument is a pointer to an unsigned short integer value returning the current angle / stroke value read from the lower channel (1/3).

valueB

This argument is a pointer to an unsigned short integer value returning the current angle / stroke value read from the higher channel (2/4).

statusA

This argument is a pointer to an unsigned short integer value returning the current status of the lower channel. The status is an ORed value of the following flags:

Flag	Description
TPMC151_STAT_QUAD	Quadrant-Detection Error detected, , data is not valid. Possible causes are: - 180° jump - Excessive rotation speed - Excessive offsets - Excessive phase shifts (> 45°)
TPMC151_STAT_LOS	SIN/COS Loss-of-Signal, SIN and COS signal is below 1/16 of the selected input range.
TPMC151_STAT_CLIP_SIN	SIN Clipping, data is not valid.
TPMC151_STAT_CLIP_COS	COS Clipping, data is not valid.
TPMC151_STAT_EXC_LOW	Excitation frequency is too low.
TPMC151_STAT_EXC_HIGH	Excitation frequency is too high.
TPMC151_STAT_LOF	RDC entered low frequency mode.
TPMC151_STAT_INIT_DONE	RDC initialization done & output data is valid.
TPMC151_STAT_AMP_OT	The excitation amplifier signaled an overtemperature condition.

statusB

This argument is a pointer to an unsigned short integer value returning the current status of the higher channel. The status is an ORed value of the flags listed above for *statusA*.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned short     valueA;
unsigned short     statusA;
unsigned short     valueB;
unsigned short     statusB;

…
```

```
…

/*-----------------------------------------------------------------------
  Read the current angle / stroke value  and status of channel pair 3/4
  -----------------------------------------------------------------------*/
result = tpmc151ReadAngleStatus( hdl, 3, &valueA, &valueB,
                                         &statusA, &statusB );
if (result == TPMC151_OK)
{
     /* handle error */
}

/* function succeeded */
printf("(Lower Chan)  Angle/Stroke = %d – Status: %04X\n", valueA,
statusA);
printf("(Higher Chan) Angle/Stroke = %d – Status: %04X\n", valueB,
statusB);
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.5 tpmc151ReadRingBuffer

NAME

tpmc151ReadRingBuffer – Read the collected data from the ring buffer of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadRingBuffer
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    int                 bufferSize,
    int                 *trigIdx,
    unsigned short      *value,
    short               *velocity,
    unsigned short      *status
)
```

DESCRIPTION

This function returns the collected data (angle/stroke, velocity and status) around the last triggered event for a specified channel. Before using this function, the ringbuffer functionality must be configured using *tpmc151ConfigRingBuffer().* This function will return immediately.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

bufferSize

This argument specifies the length of the buffers returning the collected data. All buffers (angle, velocity and status) must have at least space to receive the specified number of elements. The value can be set up to 4095.

trigIdx

This argument is a pointer to an integer value returning the index of the data collected at the trigger event. The trigger event and the number of pretrigger-collected data is specified in the function *tpmc151ConfigRingBuffer().*

value

This argument is a pointer to a buffer of unsigned short integer values with a size at least of *bufferSize* entries. The buffer will receive the angle / stroke values collected.

velocity

This argument is a pointer to a buffer of short integer values with a size at least of *bufferSize* entries. The buffer will receive the velocity values collected.

status

This argument is a pointer to a buffer of unsigned short integer values with a size at least of *bufferSize* entries. The buffer will receive the status values collected.
The status values are a ORed values of the following flags:

Flag	Description
TPMC151_VEL_QUAD	Quadrant-Detection Error detected, data is not valid. Possible causes are: - 180° jump - Excessive rotation speed - Excessive offsets - Excessive phase shifts (> 45°)
TPMC151_VEL_CLIP	SIN/COS Clipping, data is not valid.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
int                trgIndex;
unsigned short     value[100];
short              velocity[100];
unsigned short     status[100];
char               statusStr[200];
int                i;

/*-------------------------------------------------------
  Read the collected dat from channel 1 (read 100 samples)
  -------------------------------------------------------*/
result = tpmc151ReadRingBuffer ( hdl, 1, 100,
                                 &trgIndex, value, velocity, status );
if (result == TPMC151_OK)
{
     /* handle error */
}

…
```

…

```
/* function succeeded */
for (i = 0; i < 100; i++)
{
     if (trgIndex == i)
     {
          printf(“TRIGGER-");
     }
     else
     {
          printf(“        ");
     }
     printf("ang=%d, vel=%d, stat=0x%04X\n", value[i],
                                             velocity[i], status[i]);
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOMEM	Invalid buffer size
TPMC151_ERR_BUSY	Data collection is busy, or not triggered
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.6 tpmc151WaitAndReadRingBuffer

NAME

tpmc151WaitAndReadRingBuffer – Wait for trigger event and read the collected data from the ring buffer of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151WaitAndReadRingBuffer
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    int                 bufferSize,
    int                 timeout,
    int                 *trigIdx,
    unsigned short      *value,
    short               *velocity,
    unsigned short      *status
)
```

DESCRIPTION

This function waits for the trigger event and reads afterwards the collected data (angle, velocity and status) for a specified channel.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

bufferSize

This argument specifies the length of the buffers returning the collected data. All buffers (angle, velocity and status) must have at least space to receive the specified number of elements. The value can be set up to 4095.

timeout

This argument specifies the maximum time the function is willing to wait until the trigger event occurs. If the event does not occur within this time, the function will return an error. The time is specified in milliseconds.

trigIdx

This argument is a pointer to an integer value returning the index of the data collected at the trigger event. The trigger event and the number of pretrigger-collected data is specified in the function *tpmc151ConfigRingBuffer()*.

value

This argument is a pointer to a buffer of unsigned short integer values with a size at least of *bufferSize* entries. The buffer will receive the angle/stroke values collected.

velocity

This argument is a pointer to a buffer of short integer values with a size at least of *bufferSize* entries. The buffer will receive the velocity values collected.

status

This argument is a pointer to a buffer of unsigned short integer values with a size at least of *bufferSize* entries. The buffer will receive the status values collected.
The status values are a ORed values of the following flags:

Flag	Description
TPMC151_VEL_QUAD	Quadrant-Detection Error detected, data is not valid. Possible causes are: - 180° jump - Excessive rotation speed - Excessive offsets - Excessive phase shifts (> 45°)
TPMC151_VEL_CLIP	SIN/COS Clipping, data is not valid.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
int                trgIndex;
unsigned short     value[100];
short              velocity[100];
unsigned short     status[100];
char               statusStr[200];
int                i;

…
```

```
…

/*-------------------------------------------------------
  Read the collected data from channel 1 (read 100 samples)
  Wait up to 10 seconds
  -------------------------------------------------------*/
result = tpmc151ReadRingBuffer ( hdl, 1, 100, 10000,
                                 &trgIndex, value, velocity, status );
if (result == TPMC151_OK)
{
     /* handle error */
}

/* function succeeded */
for (i = 0; i < 100; i++)
{
     if (trgIndex == i)
     {
          printf("TRIGGER-");
     }
     else
     {
          printf("        ");
     }
     printf("ang=%d, vel=%d, stat=0x%04X\n", value[i],
                                             velocity[i], status[i]);
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOMEM	Invalid buffer size
TPMC151_ERR_BUSY	Data collection is busy, or not triggered
TPMC151_ERR_NOTSUP	Specified channel does not support this function
TPMC151_ERR_TIMEDOUT	The event has not occurred in the specified time

2.2.7 tpmc151ConfigRingBuffer

NAME

tpmc151ConfigRingBuffer – configures the behavior of the ring buffer and the event to fill it

SYNOPSIS

```
TPMC151_STATUS tpmc151ConfigRingBuffer
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned int        triggerMode,
    unsigned int        dataDivMode,
    unsigned int        triggerValue,
    unsigned int        pretriggerCount,
    unsigned int        flags
)
```

DESCRIPTION

This function configures the behavior of the ring buffer and the event to fill it.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

triggerMode

This argument specifies the edge or direction, the *triggerValue* has to be crossed. The following values are defined:

triggerMode	**Description**
TPMC151_TRIGEDGE_DISABLED	Disable ring buffer function
TPMC151_TRIGEDGE_LOW2HIGH	Trigger on low to high transition of the specified *triggerValue*.
TPMC151_TRIGEDGE_HIGH2LOW	Trigger on high to low transition of the specified *triggerValue*.
TPMC151_TRIGEDGE_ANY	Trigger on any transition of the specified *triggerValue*.

dataDivMode

This argument specifies the data divider mode. This mode allows to collect data over a longer time, but only using every nth sample. The following value are defined:

dataDivMode	Description
TPMC151_DATADIV_1	Store every sample.
TPMC151_DATADIV_2	Store every 2nd sample.
TPMC151_DATADIV_4	Store every 4th sample.
TPMC151_DATADIV_8	Store every 8th sample.

triggerValue

This argument specifies the angle value which must be crossed to trigger the sample collection. The *triggerMode* specifies the trigger direction.

pretriggerCount

This parameter specifies the number of samples that shall be kept before the trigger occurred. This allows to get values before the trigger has occurred. The value can be set between 0 and 4095.

flags

This argument specifies special behaviour of the ring buffer handling. The following values are defined:

Flags	Description
TPMC151_TRIG_ONESHOT	Data will be collected when the trigger occurs, data will be collected. After reading the data, the trigger will not be active again.
TPMC151_TRIG_CONTINUE	Data will be collected when the trigger occurs, data will be collected. After reading the data, the trigger is active again and will collect data with the next trigger event.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

…
```

…

```
/*-----------------------------------------
  Configure the Ring Buffer function
      Sample on any crossing of angle value 0x1000
      Store every sample
      Keep 20 samples before the trigger event
      Take just data for the first trigger event
  -----------------------------------------*/
result = tpmc151ConfigRingBuffer( hdl, 1, TPMC151_TRIGEDGE_ANY,
                                  TPMC151_DATADIV_1, 0x1000, 20,
                                  TPMC151_TRIG_ONESHOT);
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.8 tpmc151EnableExcitation

NAME

tpmc151EnableExcitation – This function sets up the excitation output of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151EnableExcitation
(
    TPMC151_HANDLE    hdl,
    int               channel,
    int               voltage,
    int               frequency
)
```

DESCRIPTION

This function sets up and enables the excitation output with individual parameters matching the connected sensor. This function is only available for resolver channels.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

voltage

This argument specifies the excitation voltage. The voltage must be specified in mV.

frequency

This parameter specifies the excitation frequency. The frequency is specified in Hz.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*-----------------------------------------------
  Setup excitation of channel 2
      Excitation voltage = 7 V
      Excitation frequency = 10 kHz
  -----------------------------------------------*/
result = tpmc151EnableExcitation ( hdl, 2, 7000, 10000);
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.9 tpmc151DisableExcitation

NAME

tpmc151DisableExcitation – This function disables the excitation output of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151DisableExcitation
(
    TPMC151_HANDLE      hdl,
    int                 channel
)
```

DESCRIPTION

This function disables the excitation output for the connected sensor. This function is only available for resolver channels.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

…
```

…

```
/*-----------------------------------------
  Disable excitation of channel 2
  -----------------------------------------*/
result = tpmc151DisableExcitation ( hdl, 2);
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.10 tpmc151ResetAFE

NAME

tpmc151ResetAFE – This function resets the analog front end

SYNOPSIS

```
TPMC151_STATUS tpmc151ResetAFE
(
     TPMC151_HANDLE       hdl
)
```

DESCRIPTION

This function resets the analog front end.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*-----------------------------------------------
  Reset the analog front end
  -----------------------------------------------*/
result = tpmc151ResetAFE ( hdl );
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_TIMEDOUT	The reset timed out

2.2.11 tpmc151ConfigInputRanges

NAME

tpmc151ConfigInputRanges – This function configures input ranges of the input interfaces for a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151ConfigInputRanges
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned int        sinRange,
    unsigned int        cosRange
)
```

DESCRIPTION

This function configures the allowed range of the input signals. The specified range must be above the maximum voltage of the input signals.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

sinRange

This argument specifies the input range of the SIN / La signal. The following ranges are defined:

range	Description
TPMC151_INRNG_14V14	14.14 V_{RMS} (default)
TPMC151_INRNG_8V84	8.84 V_{RMS}
TPMC151_INRNG_7V07	7.07 V_{RMS}
TPMC151_INRNG_3V54	3.54 V_{RMS}
TPMC151_INRNG_28V00	28 V_{RMS} (only for synchro channels)
TPMC151_INRNG_14V00	14 V_{RMS} (only for synchro channels)

cosRange

This argument specifies the input range of the COS / Lb signal. The following ranges are defined:

range	Description
TPMC151_INRNG_14V14	14.14 V_{RMS} (default)
TPMC151_INRNG_8V84	8.84 V_{RMS}
TPMC151_INRNG_7V07	7.07 V_{RMS}
TPMC151_INRNG_3V54	3.54 V_{RMS}
TPMC151_INRNG_28V00	28 V_{RMS} (only for synchro channels)
TPMC151_INRNG_14V00	14 V_{RMS} (only for synchro channels)

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*-------------------------------------------------------
  Setup input voltage ranges for channel 1 to 14.14 Vrms
  -------------------------------------------------------*/
result = tpmc151ConfigInputRanges ( hdl, 1,
                                    TPMC151_INRNG_14V14,
                                    TPMC151_INRNG_14V14);
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.12 tpmc151ConfigInterface

NAME

tpmc151ConfigInterface – This function configures the interface mode of a specified channel

SYNOPSIS

```
TPMC151_STATUS tpmc151ConfigInterface
(
    TPMC151_HANDLE      hdl,
    int                 channel,
    unsigned int        intfMode
)
```

DESCRIPTION

This function configures the interface mode, adjusting the channel's behavior and features matching to the interface.

PARAMETERS

hdl

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

channel

Specifies the channel number. Valid values are 1..4.

intfMode

This argument specifies the interface mode. The following modes are defined:

mode	Description
TPMC151_MODE_RDC	RDC
TPMC151_MODE_SYNCHRO	Synchro
TPMC151_MODE_DIFFERENTIAL_LVDT	Differential LVDT (A / B)
TPMC151_MODE_RADIOMETRIC_LVDT	Ratiometric LVDT (A-B / A+B)

The value above may be extended by ORing the following mode flag.

mode flag	Description
TPMC151_MODE_LOFCTRL	Enable low frequency mode, necessary for lower frequent excitations.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*-------------------------------------------------------
  Set interface of channel 1 to RDC interface mode
  -------------------------------------------------------*/
result = tpmc151ConfigInterface ( hdl, 1, TPMC151_MODE_RDC);
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid channel number
TPMC151_ERR_NOTSUP	Specified channel does not support this function

2.2.13 tpmc151ReadChannelStatus

NAME

tpmc151ReadChannelStatus – Reads the status of all channels and resets pending status bits

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadChannelStatus
(
    TPMC151_HANDLE      hdl,
    unsigned short      chanStatus[]
)
```

DESCRIPTION

This function reads the status of all channels and resets pending status bits.

PARAMETERS

hdl

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

chanStatus

This argument specifies an array of unsigned short values. The array must have a depth of four, one value for each of the channels. The status of the channel will be returned in the array where index 0 returns the status of channel 1, index 1 that of channel 2 and so on. The following status bits are defined and will be returned as an Ored value:

Status	Description
TPMC151_STAT_QUAD	Quadrant-Detection Error detected, data is not valid. Possible causes are: - 180° jump - Excessive rotation speed - Excessive offsets - Excessive phase shifts (> 45°)
TPMC151_STAT_LOS	SIN/COS Loss-of-Signal, SIN and COS signal is below 1/16 of the selected input range.
TPMC151_STAT_CLIP	SIN/COS Clipping, data is not valid.
TPMC151_STAT_EXC_LOW	Excitation frequency is too low.
TPMC151_STAT_EXC_HIGH	Excitation frequency is too high.
TPMC151_STAT_LOF	RDC entered low frequency mode.
TPMC151_STAT_INIT_DONE	RDC initialization done & output data is valid.
TPMC151_STAT_AMP_OT	The excitation amplifier signaled an overtemperature condition.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
int                chan;
unsigned short     status[4];
char               statusStr[200];
unsigned short     state;

/*-----------------------------------------------
  Read the current angle and status of channel 1
  -----------------------------------------------*/
result = tpmc151ReadChannelStatus( hdl, status );
if (result == TPMC151_OK)
{
     /* handle error */
}

…
```

…

```
/* function succeeded */
for (chan = 0; chan < 4; chan++)
{
     state = status[chan];
     sprintf(statusStr, "");
     if (state & TPMC151_STAT_QUAD)
          sprintf(statusStr, "%s QUAD", statusStr);
     if (state & TPMC151_STAT_LOS)
          sprintf(statusStr, "%s LOS", statusStr);
     if (state & TPMC151_STAT_CLIP)
          sprintf(statusStr, "%s CLIP", statusStr);
     if (state & TPMC151_STAT_EXC_LOW)
          sprintf(statusStr, "%s EXC_LOW", statusStr);
     if (state & TPMC151_STAT_EXC_HIGH)
          sprintf(statusStr, "%s EXC_HIGH", statusStr);
     if (state & TPMC151_STAT_LOF)
          sprintf(statusStr, "%s LOF", statusStr);
     if (state & TPMC151_STAT_INIT_DONE)
          sprintf(statusStr, "%s INIT_DONE", statusStr);
     if (state & TPMC151_STAT_AMP_OT)
          sprintf(statusStr, "%s AMP_OT", statusStr);
     printf("#%d – %s\n", chan + 1, statusStr);
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid

2.2.14 tpmc151WaitInputEvent

NAME

tpmc151WaitInputEvent – Wait for specified input events

SYNOPSIS

```
TPMC151_STATUS tpmc151WaitInputEvent
(
    TPMC151_HANDLE      hdl,
    unsigned int        eventFlags,
    int                 timeout,
    unsigned int        *occurredEvents
)
```

DESCRIPTION

This function waits for at least one of the specified events and returns the occurred events.

PARAMETERS

hdl

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

eventFlags

This argument specifies a value representing event flags for events to wait for. A channel (1..4) must be specified as parameter to the flag definition.
The parameter is an Ored value of event flags depending on the channel. The following event flags are defined:

Event Flag	Description
TPMC151_EV_FLAG_QUAD(<channel>)	Wait until a Quadrant-Detection Error is detected.
TPMC151_EV_FLAG_LOS(<channel>)	Wait for SIN/COS Loss-of-Signal, SIN and COS signal is below 1/16 of the selected input range.
TPMC151_EV_FLAG_CLIP(<channel>)	Wait for SIN/COS Clipping.
TPMC151_EV_FLAG_AMP_OT(<channel>)	Wait for an overtemperature condition of the signal amplifier.

timeout

This argument specifies the maximum time the function is willing to wait until an event occurs. If none of the specified events occur within this time, the function will return an error. The time is specified in milliseconds.

occurredEvents

This argument is a pointer to an unsigned integer value returning the occurred events. The returned value is a masked value of the *eventFlags* parameter showing the events that have occurred.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned short     eventFlags;
unsigned short     eventsOccurred;

/*-------------------------------------------------
  Wait for an event on channel 3, timout after 10 s
  -------------------------------------------------*/
eventFlags = (TPMC151_EV_FLAG_QUAD(3) | TPMC151_EV_FLAG_LOS(3) |
              TPMC151_EV_FLAG_CLIP(3) | TPMC151_EV_FLAG_AMP_OT(3));
result = tpmc151WaitInputEvent( hdl, eventFlags, 10000, &eventsOccurred);
if (result == TPMC151_OK)
{
     /* handle error */
}
printf(“Occurred Events Mask: %04X\n”, eventsOccurred);

…

/*-------------------------------------------------------
  Wait for a LOS event on all channels, timout after 60 s
  -------------------------------------------------------*/
eventFlags  = TPMC151_EV_FLAG_LOS(1);
eventFlags |= TPMC151_EV_FLAG_LOS(2);
eventFlags |= TPMC151_EV_FLAG_LOS(3);
eventFlags |= TPMC151_EV_FLAG_LOS(4);
result = tpmc151WaitInputEvent( hdl, eventFlags, 10000, &eventsOccurred);
if (result == TPMC151_OK)
{
     /* handle error */
}
printf(“Occurred Events Mask: %04X\n”, eventsOccurred);
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter values, e.g. invalid event flags
TPMC151_ERR_BUSY	The maximum number of waiting tasks has been exceeded.
TPMC151_ERR_TIMEDOUT	The event has not occurred in the specified time

2.2.15 tpmc151ReadTimer

NAME

tpmc151ReadTimer – Reads the current value of the timer counter

SYNOPSIS

```
TPMC151_STATUS tpmc151ReadTimer
(
      TPMC151_HANDLE      hdl,
      unsigned int        *timerValue
)
```

DESCRIPTION

This function reads the current value of the timer counter.

PARAMETERS

hdl

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

timerValue

This argument is a pointer to an unsigned integer value returning the current count of the counter timer.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned int       count;

…
```

```
…

/*-----------------------------------
  Read the current count of the timer
  -----------------------------------*/
result = tpmc151ReadTimer( hdl, &count );
if (result == TPMC151_OK)
{
     /* handle error */
}

Printf("The current count is: %d\n", count);
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid

2.2.16 tpmc151ConfigTimer

NAME

tpmc151ConfigTimer – configures the timer counter

SYNOPSIS

```
TPMC151_STATUS tpmc151ConfigTimer
(
    TPMC151_HANDLE      hdl,
    unsigned int        timeBaseMode,
    unsigned int        timerPreload
)
```

DESCRIPTION

This function configures the timer counter.

PARAMETERS

hdl

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

timeBaseMode

This argument specifies special time base of the counter. This allows a high variability for cycle times and resolution. The following time bases are defined:

Time Base	Description
TPMC151_TIMEBASE_100NS	Time base is 100 ns.
TPMC151_TIMEBASE_1US	Time base is 1 µs
TPMC151_TIMEBASE_1MS	Time base is 1 ms
TPMC151_TIMEBASE_1S	Time base is 1 s

timerPreload

This argument specifies the preload value. This value specifies length of one counter cycle.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*-----------------------------------------------------
  Configure counter timer with a cycle time of 1.5 s
  -----------------------------------------------------*/
result = tpmc151ConfigTimer( hdl, TPMC151_TIMEBASE_1MS, 1500 );
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_INVAL	Invalid parameter

2.2.17 tpmc151EnableTimer

NAME

tpmc151EnableTimer – enables and starts the timer counter

SYNOPSIS

```
TPMC151_STATUS tpmc151EnableTimer
(
     TPMC151_HANDLE       hdl
)
```

DESCRIPTION

This function starts the timer counter.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*--------------------
  Enable counter timer
  --------------------*/
result = tpmc151EnableTimer( hdl );
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid

2.2.18 tpmc151DisableTimer

NAME

tpmc151DisableTimer – disables and stops the timer counter

SYNOPSIS

```
TPMC151_STATUS tpmc151DisableTimer
(
     TPMC151_HANDLE       hdl
)
```

DESCRIPTION

This function stops the timer counter.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

/*------------------
  Stop counter timer
  ------------------*/
result = tpmc151DisableTimer( hdl );
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid

2.2.19 tpmc151WaitTimer

NAME

tpmc151WaitTimer – wait for next timer event

SYNOPSIS

```
TPMC151_STATUS tpmc151WaitTimer
(
    TPMC151_HANDLE      hdl;
    int                 timeout
)
```

DESCRIPTION

This function waits until a timer event occurs.

PARAMETERS

hdl

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

timeout

This argument specifies the maximum time the function is willing to wait until an event occurs. If NO event occurs within this time, the function will return an error. The time is specified in milliseconds.

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;

…
```

…

```
/*-----------------------------------------
  Wait for timer event, timeout after 5 seconds
  -----------------------------------------*/
result = tpmc151WaitTimer( hdl, 5000 );
if (result == TPMC151_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC151_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC151_ERR_INVALID_HANDLE	The specified TPMC151_HANDLE is invalid
TPMC151_ERR_BUSY	Another task/process is already using this timer
TPMC151_ERR_TIMEDOUT	The event has not occurred in the specified time

3 Appendix

3.1 Converting returned data values

The API-function will always return data values as raw unsigned short value. We have implemented macros in tpmc151api.h that help converting the raw values into angles in degree or into LVDT values.

Both macros return a double value containing the angle in degrees (0° ... 360°) or an LVDT value (-100% … +100%).

The macro for the conversion into degrees is:
`TPMC151_VALUE2DEGREES(__val__)`

and the macro for the conversion into an LVDT value is:
`TPMC151_VALUE2LVDTPERC(__val__)`

EXAMPLE

```
#include “tpmc151api.h”

TPMC151_HANDLE     hdl;
TPMC151_STATUS     result;
unsigned short     value;
unsigned short     status;
char               statusStr[200];

/*-----------------------------------------------
  Read the current value and status of channel 1
  -----------------------------------------------*/
result = tpmc151ReadValueStatus( hdl, 1, &value, &status );
if (result == TPMC151_OK)
{
     /* handle error */
}

/* function succeeded */
printf("Value = %d – Status: %04X\n", value, status);
printf("      (degrees) = %7.3f°\n", TPMC151_VALUE2DEGREES(value);
printf("      (LVDT)    = %7.3f%%\n", TPMC151_VALUE2LVDTPERC(value);
```

3.2 Build Error Message String

NAME

tpmc151ErrorMessage – build error message string

SYNOPSIS

```
char* tpmc151ErrorMessage
(
        TPMC151_STATUS          status
)
```

DESCRIPTION

This function returns a character string containing the TPMC151 error message of the specified status.

PARAMETERS

status

This argument specifies the error code returned from the TPMC151 device driver function.

RETURN VALUE

Returns a null-terminated character string containing the error code name.

3.3 Enable RTP-Support

Using TPMC151 devices tunneled from Real Time Processes (RTPs) is implemented. For this the “TEWS TPMC151 IOCTL command validation” must be enabled in system configuration.

The API source file “tpmc151api.c” must be added to the RTP-Project directory and built together with the RTP-application.

The definition of TVXB_RTP_CONTEXT must be added to the project, which is used to eliminate kernel headers, values and functions from the used driver files.

Find more detailed information in “TEWS Technologies VxWorks Device Drivers - Installation Guide”.

3.4 Debugging and Diagnostic

The TPMC151 device driver provides a function and debug statements to display versatile information of the driver installation and status on the debugging console.

If the VxBus driver is used, the TPMC151 show routine is included in the driver by default and can be called from the VxWorks shell. If this function is not needed or program space is limited the function can be removed from the code by un-defining the macro INCLUDE_TPMC151_SHOW in tpmc151drv.c

The tpmc151Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TPMC151 modules and device statistics.

If TPMC151 modules were probed but no devices were created it may be helpful to enable debugging code inside the driver code by defining the macro TPMC151_DEBUG in tpmc151drv.c.

```
tpmc151Show
Probed Modules:
    [0] TPMC151-10: Bus=4, Dev=2, DevId=0x0097, VenId=0x1498, Init=OK,
vxDev=0xffff800000140710

Associated Devices:
    [0] TPMC151-10: /tpmc151/0
                #1: Resolver or LVDT/RVDT-to-Digital Converter
                #2: Resolver or LVDT/RVDT-to-Digital Converter
                #3: Resolver or LVDT/RVDT-to-Digital Converter
                #4: Resolver or LVDT/RVDT-to-Digital Converter

Device Statistics:
    /tpmc151/0:
        open count = 0
value = 23 = 0x17
```