The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC160-SW-82

Linux Device Driver

Automotive Sensor Simulator

Version 1.0.x

User Manual

Issue 1.0.0

March 2025

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

TPMC160-SW-82

Linux Device Driver

Automotive Sensor Simulator

Supported Modules:
TPMC160

Issue	Description	Date
1.0.0	First Issue	March 6, 2025

Table of Contents

1 Introduction

The TPMC160-SW-82 Linux device driver allows the operation of the TPMC160 compatible devices conforming to the Linux I/O system specification.

The TPMC160-SW-82 device driver supports the following features:

- configure sensor channel for specific protocols
- write sensor output values
- configure cycle counter functionality
- read cycle counter values and wait for match events
- wait for PSI5 sync and status interrupt events

The TPMC160-SW-82 supports the modules listed below:

TPMC160	Automotive Sensor Simulator	PMC

In this document all supported modules and devices will be called TPMC160. Specials for a certain device will be advised.

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

TPMC160 User Manual

2 Installation

The directory TPMC160-SW-82 on the distribution media contains the following files:

TPMC160-SW-82-1.0.0.pdf	This manual in PDF format
TPMC160-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
ChangeLog.txt	Release history
Release.txt	Information about the Device Driver Release

The GZIP compressed archive TPMC160-SW-82-SRC.tar.gz contains the following files and directories:

Directory path 'tpmc160':

tpmc160.c	Driver source code
tpmc160def.h	Driver include file
tpmc160.h	Driver include file for application program
Makefile	Device driver make file
makenode	Script for device node creation
api/tpmc160api.h	API include file
api/tpmc160api.c	API source file
COPYING	Copy of the GNU Public License (GPL)
example/tpmc160exa.c	Example application
example/Makefile	Example application makefile
include/tpmodule.c	Driver independent library
include/tpmodule.h	Driver independent library header file
include/config.h	Driver independent library header file
include/tpxxxhwdep.h	HAL library header file
include/tpxxxhwdep.c	HAL library source file

In order to perform an installation, extract all files of the archive TPMC160-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC160-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tdrv018.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

- To update the device driver's module dependencies, enter:

  # **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

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 tpmc160drv

- 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 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 TPMC160 device found. The first TPMC160 device can be accessed with device node /dev/tpmc160_0, the second module with device node /dev/tpmc160_1 and so on.

The assignment of device nodes to physical TPMC160 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 tpmc160drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc160_x 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 "*tpmc160drv: 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

This paragraph is only for Linux kernels without dynamic device file system installed. The TPMC160 driver uses dynamic allocation of major device numbers per default. If this isn't suitable for the application it is possible to define a major number for the driver.

To change the major number, edit the file tpmc160def.h, change the following symbol to appropriate value, and enter `make install` to create a new driver.

TPMC160_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC160_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 tpmc160Open

NAME

tpmc160Open – opens a device.

SYNOPSIS

```
TPMC160_HANDLE tpmc160Open
(
        char            *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device 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. The following device naming must be used:

Device Number	Device Name
1	/dev/tpmc160_0
2	/dev/tpmc160_1

EXAMPLE

```
#include “tpmc160api.h”

TPMC160_HANDLE   hdl;

/*
** open the specified device
*/
hdl = tpmc160Open(“/dev/tpmc160_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 tpmc160Close

NAME

tpmc160Close – closes a device.

SYNOPSIS

```
TPMC160_STATUS tpmc160Close
(
      TPMC160_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 “tpmc160api.h”

TPMC160_HANDLE   hdl;
TPMC160_STATUS   result;

/*
** close the device
*/
result = tpmc160Close(hdl);
if (result != TPMC160_OK)
{
  /* handle close error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 tpmc160GetPciInfo

NAME

tpmc160GetPciInfo – get information of the module PCI header

SYNOPSIS

```
TPMC160_STATUS tpmc160GetPciInfo
(
    TPMC160_HANDLE          hdl,
    TPMC160_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 TPMC160_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;
} TPMC160_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 “tpmc160api.h”

TPMC160_HANDLE       hdl;
TPMC160_STATUS       result;
TPMC160_PCIINFO_BUF  pciInfoBuf

/*
** get module PCI information
*/
result = tpmc160GetPciInfo( hdl, &pciInfoBuf );

if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURN VALUE

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.4 tpmc160GetBoardInfo

NAME

tpmc160GetBoardInfo – get information of the board

SYNOPSIS

```
TPMC160_STATUS tpmc160GetBoardInfo
(
    TPMC160_HANDLE      hdl,
    unsigned int        *firmwareId,
    int                 *temperature,
    int                 *temperatureDec,
    unsigned int        *sensorAlarms
)
```

DESCRIPTION

This function returns information about the module, e.g. firmware id (version) and the core temperature.

PARAMETERS

hdl

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

firmwareId

This argument points to an unsigned 32-bit buffer, where firmware id (version) will be copied to.

temperature

This argument points to a 32-bit buffer, where the current core temperature of the FPGA (as full °C) will be stored to.

temperatureDec

This argument points to a 32-bit buffer, where the current core temperature of the FPGA (only decimal places in1/1000 °C) will be stored to.

sensorAlarms

This argument points to a 32-bit buffer, where the current announced sensor alarm flags will be stored to.

EXAMPLE

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
unsigned int       firmId;
int                healthTemp;
unsigned int       alarms;

/*
** get module board information
*/
result = tpmc160GetBoardInfo( hdl, &firmId, &healthTemp, &alarms );
if (result != TPMC160_OK)
{
  /* handle error */
}

printf(“Firmware-ID: %02d.%02d.%02d Build: %02d.\n”,
  (firmId >> 24) & 0xFF,
  (firmId >> 16) & 0xFF,
  (firmId >> 8) & 0xFF,
  firmId & 0xFF);
printf(“Temperature: %4d°C\n”, healthTemp);
printf(“Alarm-Flags: %0X\n”, alarms);
```

RETURN VALUE

On success, TDRV018_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid

3.2 Channel Configuration Functions

3.2.1 tpmc160SetMode

Name

tpmc160SetMode – configure channel mode and setup current levels

Synopsis

```
TPMC160_STATUS tpmc160SetMode
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        mode,
    int                 currentLow,
    int                 currentMid,
    int                 currentHigh
)
```

Description

This function configures the channel mode and initializes the current levels used on the channel.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

mode

This argument specifies the mode the channel be used for. The following values are defined:

Mode	Description	used current levels		
		low	mid	high
TPMC160_MODE_DISABLED	Disable channel	no	no	no
TPMC160_MODE_CUSTOM	Use custom protocol & manual current setting	yes	yes	yes
TPMC160_MODE_SQUAREWAVE	Use square wave protocol	yes	no	yes
TPMC160_MODE_PWM	Use PWM protocol	yes	no	yes
TPMC160_MODE_AK	Use AK / VD protocol	yes	yes	yes
TPMC160_MODE_PSI5	Use PSI5 protocol	yes	no	yes

currentLow

This argument specifies the current level for a low signal. The value is specified in mA.

If this current level is not used for the selected mode, the value will be ignored.

currentMid

This argument specifies the current level for a mid signal. The value is specified in mA.

If this current level is not used for the selected mode, the value will be ignored.

currentHigh

This argument specifies the current level for a high signal. The value is specified in mA.

If this current level is not used for the selected mode, the value will be ignored.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** set mode of channel #2
**     - PWM protocol
**     - Low current level: 7mA
**     - Mid current level: not used
**     - High current level: 14mA
*/
result = tpmc160SetMode ( hdl, 2, TPMC160_MODE_PWM, 7, 0, 14 );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)
TPMC160_ERR_LIMIT	Specified value is out of limits (e.g. current)

3.2.2 tpmc160SetCurrentLevel

Name

tpmc160SetCurrentLevel – Change a specified current level for a specified channel

Synopsis

```
TPMC160_STATUS tpmc160SetMode
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        levelSel,
    int                 currentLev
)
```

Description

This function changes a specified current level for a specified channel while the channel is enabled.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

levelSel

This argument specifies the current level that shall be changed. The level will be change, although the selected level is not used for the configured protocol. The following values are defined:

Current Level Selection	Description
TPMC160_CURLEV_LOW	Selects the current level for a low signal to be changed.
TPMC160_CURLEV_MID	Selects the current level for a mid signal to be changed.
TPMC160_CURLEV_HIGH	Selects the current level for a high signal to be changed.

currentLev

This argument specifies the new current level for the selected signal. The value is specified in mA.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** set low current level of channel #3
**     - current level: 17mA
*/
result = tpmc160SetCurrentLevel ( hdl, 3, TPMC160_CURLEV_LOW, 17 );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)
TPMC160_ERR_LIMIT	Specified value is out of limits (e.g. current)

3.3 Custom Protocol Functions

3.3.1 tpmc160CustomConfig

Name

tpmc160CustomConfig – configure custom protocol parameters

Synopsis

```
TPMC160_STATUS tpmc160CustomConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        cycleTime,
    unsigned int        cycleTimeBase,
    unsigned int        triggerMode,
    unsigned int        bitWidth,
    unsigned int        bitWidthBase,
    unsigned int        defaultCurrentLevel
)
```

Description

This function configures the parameters for custom protocol of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

cycleTime

This parameter specifies the cycle time. The resulting cycle time is calculated by multiplication of the base and the time value (cycleTime * cycleTimeBase).

cycleTimeBase

This argument specifies the time base for cycle time calculation. The following values are defined:

Cycle Time Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

triggerMode

This argument specifies the trigger mode. The trigger starts output from data in the FIFO. The following values are defined:

Custom Trigger Mode	Description
TPMC160_CUSTOMTRIG_MANUAL	Manual trigger mode
TPMC160_CUSTOMTRIG_SEQUENCER	Sequencer trigger mode.

bitWidth

This parameter specifies the bit width. The resulting bit width is calculated by multiplication of the bit width base and the bitWidth value (bitWidth * bitWidthBase).

bitWidthBase

This argument specifies the time base for bit width calculation. The following values are defined:

Bit Width Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

defaultCurrentLevel

This parameter specifies the default current level. cycle time. The following values are defined:

Current Level Selection	Description
TPMC160_CURLEV_OFF	Disable current.
TPMC160_CURLEV_LOW	Selects the current level for a low signal to be changed.
TPMC160_CURLEV_MID	Selects the current level for a mid signal to be changed.
TPMC160_CURLEV_HIGH	Selects the current level for a high signal to be changed.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** configure custom protocol on channel #2
**      - cycletime: 100us
**      - manual trigger mode
**      - bit width: 5us
**      - default current is low level
*/
result = tpmc160CustomConfig ( hdl, 2,
                               100, TPMC160_CYCBASE_1US,
                               TPMC160_CUSTOMTRIG_MANUAL,
                               5, TPMC160_CYCBASE_1US,
                               TPMC160_CURLEV_LOW );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)
TPMC160_ERR_LIMIT	Specified value is out of limits (e.g. cycleTime)

3.3.2 tpmc160CustomTrigger

Name

tpmc160CustomTrigger – trigger output manually in custom mode

Synopsis

```
TPMC160_STATUS tpmc160CustomTrigger
(
      TPMC160_HANDLE          hdl,
      unsigned int            channel
)
```

Description

This function triggers the output 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** trigger output on channel #2
*/
result = tpmc160CustomTrigger ( hdl, 2);
if (result != TPMC160_OK)
{
   /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.3.3 tpmc160CustomWrite

Name

tpmc160CustomWrite – write data to custom protocol FIFO

Synopsis

```
TPMC160_STATUS tpmc160CustomConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        *dataBuf,
    unsigned int        validBits,
    unsigned int        *writtenBits
)
```

Description

This function writes the specified data to the custom protocol output FIFO. The data will be send starting dependent to the selected trigger mode, immediately or after triggering the output.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

dataBuffer

This parameter points to a buffer containing the output data. The length of the array buffer is limited to 16 32-bit values.

The data is stuffed in 32-bit values containing 16 protocol bits. The first protocol bit must be filled into the two LSB bis of the dataBuffer word, bits 2 and 3 must be filled with the next protocol bit and so on. The filled data words will written in the direction of an increasing array index count.

Example of a filled dataBuffer with 33 protocol bits:

index	**bit-31**	**bit-30**	**…**	**bit-3**	**bit-2**	**bit-1**	**bit-0**
0	protocol bit 15		…	protocol bit 1		protocol bit 0	
1	protocol bit 31		…	protocol bit 17		protocol bit 16	
2	---		…	---		protocol bit 32	

The following protocol bit values are defined:

Protocol Bit Level	Description
TPMC160_CURLEV_OFF	Current is disabled for bit
TPMC160_CURLEV_LOW	Bit is low level signal.
TPMC160_CURLEV_MID	Bit is mid level signal.
TPMC160_CURLEV_HIGH	Bit is high level signal.

validBits

This parameter specifies the number of valid protocol data bits (each 2-bit) in dataBuffer.

writtenBits

This parameter points to a 32 bit buffer where the number of actually sent bits will be returned.

If the FIFO is filled before all data is sent, the value will be less than validBits.

If all data is sent, the value will be same or equal to validBits. A value greater than validBits shows that extra bits are sent completing a partially filled FIFO data word of 16-protocol bits.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
unsigned int       buffer[1];
unsigned int       written;

/*
** configure write in custom protocol on channel #2
**     Output Data: LOW,LOW,MID,HIGH,LOW
*/
buffer[0] =   TPMC160_CURLEV_LOW |
       (TPMC160_CURLEV_LOW << 2) |
       (TPMC160_CURLEV_MID << 4) |
       (TPMC160_CURLEV_HIGH << 6) |
       (TPMC160_CURLEV_LOW << 8);
result = tpmc160CustomWrite ( hdl, 2, buffer, 5, &written);
if (result != TPMC160_OK)
{
  /* handle error */
}

printf(“%d Bits written\n”, written);
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.3.4 tpmc160CustomSetDefaultCurrent

Name

tpmc160CustomSetDefaultCurrent – set the default current level

Synopsis

```
TPMC160_STATUS tpmc160CustomSetDefaultCurrent
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        levelSel
)
```

Description

This function sets the default current level for custom protocol of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

levelSel

This parameter specifies the new default current level. cycle time. The following values are defined:

Current Level Selection	Description
TPMC160_CURLEV_OFF	Disable current.
TPMC160_CURLEV_LOW	Selects the current level for a low signal to be changed.
TPMC160_CURLEV_MID	Selects the current level for a mid signal to be changed.
TPMC160_CURLEV_HIGH	Selects the current level for a high signal to be changed.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** set default output current (MID Level) of channel #2
*/
result = tpmc160CustomSetDefaultCurrent ( hdl, 2, TPMC160_CURLEV_MID );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value

3.4 Square Wave Protocol Functions

3.4.1 tpmc160SwpConfig

Name

tpmc160SwpConfig – configure square wave protocol parameters

Synopsis

```
TPMC160_STATUS tpmc160SwpConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        cycleTime,
    unsigned int        cycleTimeBase
)
```

Description

This function configures the cycle time for square wave protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

cycleTime

This parameter specifies the cycle time. The resulting cycle time is calculated by multiplication of the base and the time value (cycleTime * cycleTimeBase).

cycleTimeBase

This argument specifies the time base for cycle time calculation. The following values are defined:

Cycle Time Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** configure square wave protocol on channel 2
**     - cycletime: 100us
*/
result = tpmc160SwpConfig ( hdl, 2, 100, TPMC160_CYCBASE_1US);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.4.2 tpmc160SwpSetHighTime

Name

tpmc160SwpSetHighTime – set the high time for square wave protocol

Synopsis

```
TPMC160_STATUS tpmc160SwpSetHighTime
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        highTime,
    unsigned int        highTimeBase
)
```

Description

This function sets the high time in the square wave protocol for 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

highTime

This parameter specifies the high time. The high time is calculated by multiplying the highTime value and the specified highTimeBase.

highTimeBase

This parameter specifies the time base for high time calculation. The following values are defined:

High Time Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** high time shall be 5 us for first channel
*/
result = tpmc160SwpSetHighTime ( hdl, 0, 5, TPMC160_CYCBASE_1US);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.4.3 tpmc160SwpSetOutput

Name

tpmc160SwpSetOutput – set square wave protocol signal

Synopsis

```
TPMC160_STATUS tpmc160SwpSetOutput
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        cycleTime,
    unsigned int        cycleTimeBase,
    unsigned int        highTimePart
)
```

Description

This function sets the output signal for square wave protocol on the specified channel. The cycle time is specified and the high time as part of a whole cycle.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

cycleTime

This parameter specifies the cycle time. The resulting cycle time is calculated by multiplication of the base and the time value (cycleTime * cycleTimeBase).

cycleTimeBase

This argument specifies the time base for cycle time calculation. The following values are defined:

Cycle Time Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

highTimePart

This parameter specifies the high time part of a full cycle. The high time is specified in $^{1}/_{1000}$ part of the cycle time.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** set square wave protocol output on channel #2
** - cycletime: 100us
** - highTime part: 500/1000 (50%)
*/
result = tpmc160SwpSetOutput ( hdl, 2, 100, TPMC160_CYCBASE_1US, 500 );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.5 PWM Functions

3.5.1 tpmc160PwmConfig

Name

tpmc160PwmConfig – configure PWM protocol parameters

Synopsis

```
TPMC160_STATUS tpmc160PwmConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        cycleTime,
    unsigned int        cycleTimeBase,
    unsigned int        tpBaseLength
)
```

Description

This function configures the parameters like cycle and pulse time for PWM protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

cycleTime

This parameter specifies the cycle time. The resulting cycle time is calculated by multiplication of the base and the time value (cycleTime * cycleTimeBase).

cycleTimeBase

This argument specifies the time base for cycle time calculation. The following values are defined:

Cycle Time Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

tpBaseLength

This argument specifies the base length for the TP generation. The length is scaled in steps of 100ns.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** configure PWM protocol on channel 2
**     - cycletime:   45us
**     - base for TP:  1.5us
*/
result = tpmc160PwmConfig ( hdl, 2, 45, TPMC160_CYCBASE_1US, 15);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.5.2 tpmc160PwmSetTPLength

Name

tpmc160PwmSetTPLength – set length of the TP for PWM protocol

Synopsis

```
TPMC160_STATUS tpmc160SetTPLength
(
      TPMC160_HANDLE        hdl,
      unsigned int          channel,
      unsigned int          tpLength
)
```

Description

This function specifies the length of the TP pulse for the PWM protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

tpLength

This parameter specifies the length of the TP pulse. The length of the TP pulse will be this value multiplied with the tpBaseLength specified in tpmc160PwmConfig().

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** set the TP pulse length to 5x tpBaseLength on channel #2
*/
result = tpmc160PwmSetTPLength ( hdl, 2, 5);
if (result != TPMC160_OK)
{
   /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)
TPMC160_ERR_LIMIT	Specified value is out of limits (e.g. tpLength)

3.6 AK/VK Functions

3.6.1 tpmc160AkVkConfig

Name

tpmc160AkVkConfig – configure AK/VK protocol parameters

Synopsis

```
TPMC160_STATUS tpmc160AkVkPwmConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        cycleTime,
    unsigned int        cycleTimeBase,
    unsigned int        tpBitWidth
)
```

Description

This function configures the parameters cycle time and bit width for AK/VK protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

cycleTime

This parameter specifies the cycle time. The resulting cycle time is calculated by multiplication of the base and the time value (cycleTime * cycleTimeBase).

cycleTimeBase

This argument specifies the time base for cycle time calculation. The following values are defined:

Cycle Time Base	Description
TPMC160_CYCBASE_50NS	Timebase is 50ns.
TPMC160_CYCBASE_100NS	Timebase is 100ns.
TPMC160_CYCBASE_1US	Timebase is 1µs.
TPMC160_CYCBASE_1MS	Timebase is 1ms.

tpBitWidth

This argument specifies the width of the TP bit. The width is scaled in steps of 100ns.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** configure AK/VK protocol on channel #2
**     - cycletime:    45us
**     - TP bit width: 5us
*/
result = tpmc160AkVkConfig ( hdl, 2, 45, TPMC160_CYCBASE_1US, 50);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.6.2 tpmc160AkVkSetControl

Name

tpmc160AkVkSetControl – set the output value for the AK/VK protocol

Synopsis

```
TPMC160_STATUS tpmc160AkVkSetControl
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        bits,
    unsigned int        numBits,
    unsigned int        speedPulse
)
```

Description

This function set the output value of the AK/VK protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

bits

This parameter specifies the output bits. The maximum length is 9.

numBits

This parameter specifies the used number of bits in bits.

speedPulse

This parameter specifies the kind of the speed pulse. The following values are defined:

Speedpulse	Description
TPMC160_SPEEDPULSE_NORMAL	Normal (High signal)
TPMC160_SPEEDPULSE_SEQUENCER	Artificial (MID signal)

Example

```
   #include “tpmc160api.h”

   TPMC160_HANDLE     hdl;
   TPMC160_STATUS     result;

   /*
   ** set new AK/VK output values on channel #2
   **     - 4 bits high, 4 bits low
   **     - use normal speed pulse
   */
result = tpmc160AkVkSetControl ( hdl, 2, 8, 0xF0,
                                 TPMC160_SPEEDPULSE_NORMAL);
   if (result != TPMC160_OK)
   {
     /* handle error */
   }
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7 PSI5 Functions

3.7.1 tpmc160Psi5Config

Name

tpmc160Psi5Config – configure PSI5 protocol parameters

Synopsis

```
TPMC160_STATUS tpmc160Psi5Config
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        cycleTime,
    unsigned int        tpBitWidth,
    unsigned int        psi5Mode,
    unsigned int        slotDelay,
    unsigned int        startbits,
    unsigned int        numDatabits,
    unsigned int        defaultFrame,
    unsigned int        flags
)
```

Description

This function configures the parameters cycle time and bit width for PSI5 protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

cycleTime

This parameter specifies the cycle time. The resulting cycle time is specified in steps of 1µs.

tpBitWidth

This argument specifies the width of the TP bit. The width is scaled in steps of 100ns.

psi5Mode

This argument specifies the PSI5 Communication Mode. The following values are defined:

Value	Description
TPMC160_PSI5_ASYNC	Asynchronous Mode
TPMC160_PSI5_SYNC	Synchronous Mode without Daisy Chain
TPMC160_PSI5_SYNC_DC	Synchronous Mode with Daisy Chain
TPMC160_PSI5_VARSYNC	Variable Sync pulse mode

slotDelay

This argument specifies the delay between the synchronization pulse and the data frame. The delay is specified in microseconds. Not used in Asynchronous Mode.

startbits

This argument specifies the values of the startbits S1 and S2 (bits 0 and 1). If no startbits shall be used, specify the corresponding flag.

numDatabits

This argument specifies the number of data bits in the frame.

defaultFrame

This argument specifies the default data frame.

flags

This argument specifies additional configuration flags. The following values are defined:

Value	Description
TPMC160_PSI5_PM_TOOTHGAP	Tooth Gap Mode (Variable Sync Pulse Mode)
TPMC160_PSI5_PM_PULSEWIDTH	Pulse Width Mode (Variable Sync Pulse Mode)
TPMC160_PSI5_NOSTARTBITS	Do not use startbits

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** configure PSI5 protocol on channel #2
**     - cycletime: 200us
**     - TP bit width: 5us
**     - ASYNC mode, Startbits S1/S2: 11
**     - 16 databits, default frame 0x0000
*/
result = tpmc160Psi5Config ( hdl,
         2,                     /* channel                 */
         200, 50,               /* cycletime & bitwidth    */
         TPMC160_PSI5_ASYNC,    /* Mode                    */
         0,                     /* slotDelay: not used     */
         (1<<1) | (1 << 0),     /* Startbits               */
         16,                    /* numDatabits             */
         0x0000,                /* Default Frame value     */
         0                      /* Flags                   */
);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7.2 tpmc160Psi5DetectConfig

Name

tpmc160Psi5DetectConfig – configure PSI5 signal detection parameters

Synopsis

```
TPMC160_STATUS tpmc160Psi5DetectConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        syncSustainVoltage,
    unsigned int        underVoltage,
    unsigned int        resetThresholdTime
)
```

Description

This function configures signal detection parameters for PSI5 protocol 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

syncSustainVoltage

This parameter is the voltage that is used to detect PSI5 SYNC pulses. This value is specified in steps of 1 mV.

underVoltage

This parameter is the voltage that is used to detect PSI5 undervoltage resets. This value is specified in steps of 1 mV.

resetThresholdTime

This argument specifies the time which initiates a reset. The time is scaled in steps of 100µs.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** configure PSI5 detection on channel 2
**     - sync Sustain voltage:    8V
**     - reset under voltage:     3V
**     - reset undervoltage time: 2ms
*/
result = tpmc160Psi5DetectConfig ( hdl, 2, 8000, 3000, 2000);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7.3 tpmc160Psi5SyncConfig

Name

tpmc160Psi5SyncConfig – Configure number of SyncBits in word

Synopsis

```
TPMC160_STATUS tpmc160Psi5SyncConfig
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        syncBits
)
```

Description

This function configures the number of sync bits within the sync word.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

syncBits

This parameter specifies the number of sync bits. Valid values are 1 to 32.

Example

```
   #include “tpmc160api.h”

   TPMC160_HANDLE     hdl;
   TPMC160_STATUS     result;

   /*
   ** configure the number of sync bits on channel #2
   **     - 16 sync bits
   */
result = tpmc160Psi5SyncConfig ( hdl, 2, 16);
   if (result != TPMC160_OK)
   {
     /* handle error */
   }
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7.4 tpmc160Psi5Write

Name

tpmc160Psi5Write – write PSI5 data to FIFO

Synopsis

```
TPMC160_STATUS tpmc160Psi5Write
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        *dataFrames,
    unsigned int        numFrames,
    unsigned int        *writtenFrames
)
```

Description

This function writes PSI5 data frames into the FIFO of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

dataFrames

This parameter specifies the pointer to a 32bit data buffer. The referenced memory must be at least of numFrames size.

numFrames

This parameter specifies the number of data frames stored at dataFrames.

writtenFrames

This parameter returns the number of data frames successfully written into the FIFO buffer.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
unsigned int       data[2];
unsigned int       written;

/*
** write data frames to channel #2
*/
data[0] = 0x11223344;
data[1] = ...;

result = tpmc160Psi5Write ( hdl, 2, data, 2, &written);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7.5 tpmc160Psi5ReadSync

Name

tpmc160Psi5ReadSync – read PSI5 Sync Data Word

Synopsis

```
TPMC160_STATUS tpmc160Psi5ReadSync
(
        TPMC160_HANDLE          hdl,
        unsigned int            channel,
        unsigned int            *syncData
)
```

Description

This function reads the PSI5 Sync data word of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

syncData

This parameter specifies the pointer to a 32bit data buffer where the Sync Data Word is returned.

Example

```
   #include “tpmc160api.h”

   TPMC160_HANDLE     hdl;
   TPMC160_STATUS     result;
   unsigned int       syncData;

   /*
   ** read sync data word of channel #2
   */
result = tpmc160Psi5ReadSync ( hdl, 2, &syncData);
   if (result != TPMC160_OK)
   {
     /* handle error */
   }
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7.6 tpmc160Psi5GetSyncError

Name

tpmc160Psi5GetSyncError – read PSI5 Sync Error Status

Synopsis

```
TPMC160_STATUS tpmc160Psi5GetSyncError
(
        TPMC160_HANDLE          hdl,
        unsigned int            channel,
        unsigned int            *syncError
)
```

Description

This function reads the PSI5 Sync Error Status of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

syncError

This parameter specifies the pointer to a 32bit data buffer where the Sync Error Status is returned. The following values are possible:

Value	Description
0	No unexpected sync pulse detected yet
1	Absence of the sync pulse at the expected time window
2	Too short for a short pulse
3	Too long for a short pulse, too short for a long pulse
4	Too long for a long pulse
5	Long pulse instead of expected short pulse detected

Example

```
   #include “tpmc160api.h”

   TPMC160_HANDLE     hdl;
   TPMC160_STATUS     result;
   unsigned int       syncError;

   /*
   ** read sync error status of channel #2
   */
result = tpmc160Psi5GetSyncError ( hdl, 2, &syncError);
   if (result != TPMC160_OK)
   {
     /* handle error */
   }
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.7.7 tpmc160WaitPsi5SyncMatch

Name

tpmc160WaitPsi5SyncMatch – wait for a PSI5 Sync Match event

Synopsis

```
TPMC160_STATUS tpmc160WaitPsi5SyncMatch
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    int                 timeout
)
```

Description

This function waits for a PSI5 Sync Match event of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

timeout

Specifies the amount of time (in milliseconds) the caller is willing to wait for the specified event to occur. The granularity is seconds. A value of 0 means wait indefinitely.

Example

```
   #include “tpmc160api.h”

   TPMC160_HANDLE     hdl;
   TPMC160_STATUS     result;
   unsigned int       syncError;

   /*
   ** Wait for a Sync Match event on channel #2
   ** Timeout: 5 seconds
   */
result = tpmc160WaitPsi5SyncMatch ( hdl, 2, 5000);
   if (result != TPMC160_OK)
   {
     /* handle error */
   }
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)
TPMC160_ERR_TIMEOUT	The requested event did not occur within the specified time.

3.7.8 tpmc160WaitPsi5Status

Name

tpmc160WaitPsi5Status – wait for a PSI5 Status event

Synopsis

```
TPMC160_STATUS tpmc160WaitPsi5Status
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        evType,
    int                 timeout
)
```

Description

This function waits for a PSI5 Status event of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

evType

This argument specifies the event type to wait for. The following values are defined:

Value	Description
TPMC160_PSI5_EVSTATUS_OFLOW	SYNC bit overflow
TPMC160_PSI5_EVSTATUS_SPUR	Spurious Sync signal
TPMC160_PSI5_EVSTATUS_RESET	Reset TTH status

timeout

Specifies the amount of time (in milliseconds) the caller is willing to wait for the specified event to occur. The granularity is seconds. A value of 0 means wait indefinitely.

Example

```
   #include “tpmc160api.h”

   TPMC160_HANDLE     hdl;
   TPMC160_STATUS     result;
   unsigned int       syncError;

   /*
   ** Wait for a PSI5 “Spurious Sync Signal” event on channel #2
   ** Timeout: 5 seconds
   */
result = tpmc160WaitPsi5Status ( hdl, 2, TPMC160_PSI5_EVSTATUS_SPUR, 5000);
   if (result != TPMC160_OK)
   {
     /* handle error */
   }
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)
TPMC160_ERR_TIMEOUT	The requested event did not occur within the specified time.

3.8 Cycle Counter Functions

3.8.1 tpmc160EnableCycleCount

Name

tpmc160EnableCycleCount – enables the cycle counter

Synopsis

```
TPMC160_STATUS tpmc160EnableCycleCount
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel
)
```

Description

This function enabled the cycle counter of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** enable cycle counter on channel #2
*/
result = tpmc160EnableCycleCount ( hdl, 2);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.8.2 tpmc160DisableCycleCount

Name

tpmc160DisableCycleCount – disables the cycle counter

Synopsis

```
TPMC160_STATUS tpmc160DisableCycleCount
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel
)
```

Description

This function disables the cycle counter of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** disable cycle counter on channel #2
*/
result = tpmc160DisableCycleCount ( hdl, 2);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.8.3 tpmc160ResetCycleCount

Name

tpmc160ResetCycleCount – resets the cycle counter

Synopsis

```
TPMC160_STATUS tpmc160ResetCycleCount
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel
)
```

Description

This function resets the cycle counter of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** reset cycle counter of channel #2
*/
result = tpmc160ResetCycleCount ( hdl, 2);
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.8.4 tpmc160GetCycleCount

Name

tpmc160GetCycleCount – get current cycle count of a specified channel

Synopsis

```
TPMC160_STATUS tpmc160GetCycleCount
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        *count
)
```

Description

This function returns the current cycle count of 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

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

count

This argument points to an unsigned int 32-bit buffer, where the current count of the channel is returned.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
unsigned int       count;

/*
** get current cycle count of channel #4
*/
result = tpmc160GetCycleCounte ( hdl, 4, & count);
if (result != TPMC160_OK)
{
  /* handle error */
}

printf(“Cycle Count: %d\n”, count);
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.8.5 tpmc160GetAllCycleCounts

Name

tpmc160GetAllCycleCounts – get cycle count values of all channels

Synopsis

```
TPMC160_STATUS tpmc160GetAllCycleCounts
(
    TPMC160_HANDLE      hdl,
    unsigned int        counts[]
)
```

Description

This function returns the current cycle counts of all channels.

Parameters

hdl

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

counts

This argument points to an array of unsigned int 32-bit values, where the current voltages will be stored to. The array size must match to the number of available channels on the selected board.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
unsigned int       counts[8];

/*
** get current cycle count of all channels
*/
result = tpmc160GetAllCycleCounts ( hdl, counts );
if (result != TPMC160_OK)
{
  /* handle error */
}

printf(“Cycle Count #0: %d uV\n”, counts[0]);
printf(“Cycle Count #1: %d uV\n”, counts [1]);
…
printf(“Cycle Count #7: %d uV\n”, counts [7]);
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid

3.8.6 tpmc160SetCycleCountMatch

Name

tpmc160SetCycleCountMatch – set the cycle counter match value

Synopsis

```
TPMC160_STATUS tpmc160SetCycleCountMatch
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    unsigned int        matchValue
)
```

Description

This function sets the cycle counter match value of the specified channel. If the cycle counter reaches this value the counter will be reset and starts counting with 0. Calling this function resets the count, too.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

matchValue

This parameter specifies the cycle counter match value.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** set cycle counter match value protocol on channel #2 to 10000
*/
result = tpmc160SetCycleCountMatch ( hdl, 2, 10000 );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.8.7 tpmc160WaitCycleCountMatch

Name

tpmc160WaitCycleCountMatch – wait for cycle counter match event

Synopsis

```
TPMC160_STATUS tpmc160WaitCycleCountMatch
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    int                 timeout
)
```

Description

This function waits for a cycle counter match event of the specified channel. If the event, cycle counter matches the cycle counter match value, occurs, the function will return. If the event does not occur in a specified time, the function will return with an appropriate error code.

Parameters

hdl

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

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

timeout

This parameter specifies the time the function is willing to wait for the event. The timeout is specified in milliseconds.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;

/*
** wait for a cycle counter match event on channel #2
**     - timeout after:  1s
*/
result = tpmc160WaitCycleCountMatch ( hdl, 2, 1000 );
if (result != TPMC160_OK)
{
  /* handle error */
}
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.9 Channel Monitoring Functions

3.9.1 tpmc160GetVoltage

Name

tpmc160GetVoltage – get current voltage of a specified channel

Synopsis

```
TPMC160_STATUS tpmc160GetVoltage
(
    TPMC160_HANDLE      hdl,
    unsigned int        channel,
    int                 *voltage
)
```

Description

This function returns the current voltage 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.

channel

This argument specifies the used channel. The first channel is selected with 0, the second with 1 and so on.

voltage

This argument points to a signed int 32-bit buffer, where the current voltage of the channel is returned. The value is scaled to µVolt.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
int                voltage;

/*
** get current voltage at channel #4
*/
result = tpmc160GetVoltage ( hdl, 4, &voltage );
if (result != TPMC160_OK)
{
  /* handle error */
}

printf(“Voltage: %d uV\n”, voltage);
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC160_ERR_INVAL	Invalid parameter value specified (e.g. channel number)

3.9.2 tpmc160GetAllVoltages

Name

tpmc160GetAllVoltages – get current voltages of all channels

Synopsis

```
TPMC160_STATUS tpmc160GetAllVoltages
(
    TPMC160_HANDLE      hdl,
    int                 voltages[]
)
```

Description

This function returns the current voltages of all channels.

Parameters

hdl

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

voltages

This argument points to an array of signed int 32-bit values, where the current voltages will be stored to. The array size must match to the number of available channels on the selected board. The values are scaled to µVolt.

Example

```
#include “tpmc160api.h”

TPMC160_HANDLE     hdl;
TPMC160_STATUS     result;
int                voltages[8];

/*
** get current voltage of all channels
*/
result = tpmc160GetAllVoltages ( hdl, voltages );
if (result != TPMC160_OK)
{
  /* handle error */
}

printf(“Voltage #0: %d uV\n”, voltages[0]);
printf(“Voltage #1: %d uV\n”, voltages[1]);
…
printf(“Voltage #7: %d uV\n”, voltages[7]);
```

RETURNS

On success, TPMC160_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC160_ERR_INVALID_HANDLE	The specified device handle is invalid

4 Diagnostic

If the TPMC160 does not work properly it is helpful to get some status information from the driver respective kernel.

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 TPMC160 driver (see also the proc man pages).

```
# lspci -v
  …
03:00.0 Signal processing controller: TEWS Technologies GmbH Device 00a0
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: medium devsel, IRQ 19
        Memory at a1100000 (32-bit, non-prefetchable) [size=4K]
        Kernel driver in use: TEWS Technologies TPMC160 Automotive Sensor Simulator
        Kernel modules: tpmc160drv
  …
```

```
# cat /proc/devices
Character devices:
  1 mem
  2 pty
  …
248 tpmc160drv
  …
```