The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC501-SW-25

Integrity Device Driver

32 Channel 16-bit ADC PMC

Version 1.1.x

User Manual

Issue 1.1.0

June 2018

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7 25469 Halstenbek, Germany
Phone: +49 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com www.tews.com

TPMC501-SW-25

Integrity Device Driver

32 Channel 16-bit ADC PMC

Supported Modules:
TPMC501

Issue	Description	Date
1.0.0	First Issue	September 5, 2012
1.1.0	Installation changed for Integrity 11 Examples changed into an interactive application	June 4, 2018

Table of Contents

1 Introduction

The TPMC501-SW-25 Integrity device driver software allows the operation of the supported PMC conforming to the Integrity I/O system specification. The software is designed and tested with Integrity 11.4.4.

The driver software uses mutual exclusion to prevent simultaneous requests by multiple tasks from interfering with each other.

The TPMC501-SW-25 device driver supports the following features:

- Execute AD conversion and read data
- Choosing gain, channel, input interface for read
- Correction of input data with board-specific calibration data
- Support of ADC sequencer mode
- Configurable sequencer cycle time, input FIFO size, and channel parameters
- Sequencer read with wait and no wait option

The TPMC501-SW-25 supports the modules listed below:

TPMC501	32(16) Channel - 16-bit ADC	(PMC)

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

TPMC501 User Manual

2 Installation

The following files are located on the distribution media:

Directory path TPMC501-SW-25:

tpmc501.c	TPMC501 device driver source
tpmc501def.h	TPMC501 driver include file
tpmc501.h	TPMC501 include file for driver and application
tpmc501api.c	Application interface, simplifies device access
tpmc501api.h	Include file for API and applications
example/tpmc501exa.c	Example application
TPMC501-SW-25-1.1.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Driver Installation

Copy the TPMC501 driver files into a desired driver or project path. The driver source file tpmc501.c must be included into the kernel project and the BSP paths must be added to the include search paths of the file. Set Options… ➔ Project → Include Directories, then double click and add the new paths:

$(__OS_DIR)/bsp
$(__OS_DIR)/system
$(__OS_DIR)/modules/ghs/bspsrc
$(__OS_DIR)/modules/ghs/bspsrc/support
$(__OS_DIR)/modules/ghs/bspsrc/driver/busspace

Afterwards the project must be rebuilt. The driver will be started automatically after booting the image and the driver will be requested if a matching device is detected in the system.

For further information building a kernel, please refer to MULTI and INTEGRITY Documentation.

2.2 TPMC501 Applications

Copy the TPMC501 API files (tpmc501api.c, tpmc501api.h, and tpmc501.h) into a desired application path, and include tpmc501api.c into the application project.

The application source file must include tpmc501api.h. If these steps are done, the TPMC501 API can be used and the devices will be accessible.

3 API Documentation

3.1 tpmc501Open

NAME

tpmc501Open() – open a device.

SYNOPSIS

```
TPMC501_HANDLE tpmc501Open
(
      char                          *DeviceName
)
```

DESCRIPTION

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

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TPMC501 device is named “tpmc501_0”, the second device is named “tpmc501_1” and so on.

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tpmc501Open(“tpmc501_0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device descriptor pointer or NULL if the function fails.

3.2 tpmc501Close

NAME

tpmc501Close() – close a device.

SYNOPSIS

```
TPMC501_STATUS tpmc501Close
(
    TPMC501_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 “tpmc501api.h”

TPMC501_HANDLE     hdl;
TPMC501_STATUS     result;

/*
** close the device
*/
result = tpmc501Close(hdl);
if (result != TPMC501_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified device handle is invalid

3.3 tpmc501GetModuleInfo

NAME

tpmc501GetModuleInfo – Get module information data

SYNOPSIS

```
TPMC501_STATUS tpmc501GetModuleInfo
(
    TPMC501_HANDLE          hdl,
    TPMC501_INFO_BUFFER     *pModuleInfo
)
```

DESCRIPTION

This function reads module information data such as configured module type, location on the PCI bus and factory programmed correction data.

PARAMETERS

hdl

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

pModuleInfo

This argument specifies a pointer to the module information buffer.

```
typedef struct
{
    unsigned int    Variant;
    unsigned int    PciBusNo;
    unsigned int    PciDevNo;
    int             ADCOffsetCal[4];
    int             ADCGainCal[4];
} TPMC501_INFO_BUFFER;
```

Variant

This parameter returns the configured module variant (e.g. 10 for a TPMC501-10).

PciBusNo, PciDevNo

These parameters specify the PCI location of this module.
(This information is not available for TPMC501-SW-25.)

ADCOffsetCal[4]

This array returns the factory programmed offset correction value for the different gains. Array index 0 contains the value for gain 1, index 1 contains the value for gain 2 and so forth.

ADCGainCal[4]

This array returns the factory programmed gain correction for the different gains. Array index 0 contains the value for gain 1, index 1 contains the value for gain 2 and so forth.

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE        hdl;
TPMC501_STATUS        result;
TPMC501_INFO_BUFFER   ModuleInfo

result = tpmc501GetModuleInfo(hdl, &ModuleInfo);
if (result != TPMC501_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.

3.4 tpmc501SetModelType

NAME

tpmc501SetModelType – configures the TPMC501 board type

SYNOPSIS

```
TPMC501_STATUS tpmc501SetModelType
(
    TPMC501_HANDLE      hdl,
    int                 modType
)
```

DESCRIPTION

This function configures the TPMC501 board type.

This function must be called after initialization of the ADC device, before any other function accesses the device.

PARAMETERS

hdl

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

modType

This parameter specifies the model type of the TPMC501. The following model types are supported:

Model Type	Description
10	TPMC501-10 --- input range +/- 10V, gains: 1,2,5,10, front panel I/O
11	TPMC501-11 --- input range +/- 10V, gains: 1,2,4,8, front panel I/O
12	TPMC501-12 --- input range 0…10V, gains: 1,2,5,10, front panel I/O
13	TPMC501-13 --- input range 0…10V, gains: 1,2,4,8, front panel I/O
20	TPMC501-20 --- input range +/- 10V, gains: 1,2,5,10, back I/O
21	TPMC501-21 --- input range +/- 10V, gains: 1,2,4,8, back I/O
22	TPMC501-22 --- input range 0…10V, gains: 1,2,5,10, back I/O
23	TPMC501-23 --- input range 0…10V, gains: 1,2,4,8, back I/O

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE     hdl;
TPMC501_STATUS     result;

/*
** tell the driver that this is a TPMC501-10
*/
result = tpmc501SetModelType(hdl, 10);
if (result != TPMC501_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC501_ERR_INVAL	Unsupported or invalid TPMC501 model type specified
TPMC501_ERR_BUSY	The device is busy

3.5 tpmc501Read

NAME

tpmc501Read – perform AD conversion and read value

SYNOPSIS

```
TPMC501_STATUS tpmc501Read
(
    TPMC501_HANDLE      hdl,
    int                 channel,
    int                 gain,
    unsigned int        flags,
    int                 *pAdcVal
)
```

DESCRIPTION

This function starts an AD conversion on a specified input channel and returns the converted value.

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 input channel. Allowed values are 1...32 for single ended interface. If a differential interface is selected (TPMC501_DIFF set in flags) the values 1…16 are allowed.

gain

This argument specifies the input gain that shall be used. Allowed values are 1, 2, 5, 10 or 1, 2, 4, 8 depending on the module type.

flags

Set of bit flags that control the AD conversion. The following flags could be OR'ed:

Flag	Meaning
TPMC501_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC501_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC501 EEPROM.
TPMC501_FAST	If this flag is set the fast (polled) mode will be used. The driver will not use interrupts, instead it will wait in a busy loop until the settling time (if necessary) and the conversion is finished. Conversions using this mode will be handled faster, but the processor executes a busy loop and other tasks will not be handled during the loops.

pAdcVal

This argument points to a buffer where the AD value will be returned.

EXAMPLE

```
#include "tpmc501api.h"

TPMC501_HANDLE     hdl;
TPMC501_STATUS     result;
int                in_value;

/*
** read AD value from channel 5 with gain = 2
** single endend input, correction enabled, use fast mode
*/
result = tpmc501Read(hdl,
                     5,
                     2,
                     TPMC501_CORR | TPMC501_FAST,
                     &in_value);
if (result != TPMC501_OK)
{
     /* handle error */
}
else
{
     printf("ADC #5: %d\n", in_value);
}
```

RETURN VALUE

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC501_ERR_BUSY	The device is busy
TPMC501_ERR_INVAL	Invalid parameter specified: invalid channel number, gain or flag specified
TPMC501_ERR_TIMEOUT	The conversion timed out
TPMC501_ERR_CONFIG	TPMC501 model type unknown or not configured

3.6 tpmc501StartSequencer

NAME

tpmc501StartSequencer – setup and start sequencer operation

SYNOPSIS

```
TPMC501_STATUS tpmc501StartSequencer
(
    TPMC501_HANDLE      hdl,
    unsigned int        CycleTime,
    unsigned int        NumOfBufferPages,
    unsigned int        NumOfChannels,
    TPMC501_CHAN_CONF   *ChanConf
)
```

DESCRIPTION

This function sets up and starts the sequencer. The setup specifies the channels to be used in sequencer mode and how they will be setup, defining gain, correction and input interface. Additionally the sequencer cycle time is defined and depth of the driver's sequencer FIFO will be configured.

PARAMETERS

hdl

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

CycleTime

This argument specifies the repeat frequency of the sequencer in 100µs steps. Each time the sequencer timer reaches the programmed cycle time a new AD conversion of all active channels is started. Valid values are in the range from 1 (100µs) to 65535 (6.5535 seconds).

NumOfBufferPages

This argument specifies the number of sample blocks in the ring buffer. A sample block contains the samples of all channels (NumOfChannels) per sequencer cycle.

NumOfChannels

This argument specifies the number of active channels for this job. The maximum number is 32.

ChanConf

This array of channel configuration structures specifies the configuration of the active channels. The channel configuration defines the channel number, the gain and some flags. The ordering of channels in a ring buffer page is the same as defined in this array.

```
typedef struct
{
    unsigned int    ChanToUse;
    unsigned int    gain;
    unsigned int    flags;
} TPMC501_CHAN_CONF;
```

ChanToUse

This parameter specifies the input channel number. Valid channels for single-ended mode are 1…32, for differential mode 1...16.

gain

This Parameter specifies the gain for this channel. Valid gains are 1, 2, 5, 10 for *TPMC501-10/-12/-20/-22* and 1, 2, 4, 8 for *TPMC501-11/-13/-21/-23.*

flags

Set of bit flags that control the AD conversion. The following flags could be OR'ed:

Flag	Meaning
TPMC501_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC501_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC501 EEPROM.

EXAMPLE

```
#include "tpmc501api.h"

TPMC501_HANDLE     hdl;
TPMC501_STATUS     result;
unsigned int       CycleTime;
unsigned int       NumOfBufferPages;
unsigned int       NumOfChannels;
TPMC501_CHAN_CONF  ChanConf[TPMC501_MAX_CHAN];

CycleTime          = 5000;
NumOfBufferPages   = 100;
NumOfChannels      = 2;

…
```

```
…

ChanConf[0].ChanToUse   = 1;
ChanConf[0].gain        = 1;
ChanConf[0].flags       = TPMC501_CORR;

ChanConf[1].ChanToUse   = 20;
ChanConf[1].gain        = 5;
ChanConf[1].flags       = TPMC501_CORR;
…

// start the sequencer
result = tpmc501StartSequencer(hdl,
                               CycleTime,
                               NumOfBufferPages,
                               NumOfChannels,
                               ChanConf);
if (result != TPMC501_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC501_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.
TPMC501_ERR_INVAL	At least one of the parameters is invalid.
TPMC501_ERR_CONFIG	TPMC501 model type unknown or not configured
TPMC501_ERR_NOMEM	Allocating buffer failed

3.7 tpmc501StopSequencer

NAME

tpmc501StopSequencer – Stop the sequencer

SYNOPSIS

```
TPMC501_STATUS tpmc501StopSequencer
(
     TPMC501_HANDLE          hdl
)
```

DESCRIPTION

This function stops execution of the sequencer mode on the specified device.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE  hdl;
TPMC501_STATUS  result;

result = tpmc501StopSequencer(hdl);
if (result != TPMC501_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified device handle is invalid

3.8 tpmc501GetDataBuffer

NAME

tpmc501GetDataBuffer – Get next data block of sequencer samples

SYNOPSIS

```
TPMC501_STATUS tpmc501GetDataBuffer
(
    TPMC501_HANDLE      hdl,
    unsigned int        flags,
    int                 *pData,
    unsigned int        *pStatus
)
```

DESCRIPTION

This function returns the next available data block in the ring buffer containing ADC data of configured sequencer channels.

If specified the function will return immediately, even if there is no data available. If the function should wait for data the function returns immediately if data is already available in FIFO or wait for sequencer cycle completion. The function timeout, if there is an abnormal delay during wait.

PARAMETERS

hdl

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

flags

Set of bit flags that control the sequencer read. The following flags could be OR'ed:

Flag	Meaning
TPMC501_NOWAIT	If this flag is set the function will return immediately, even if there is no data available. If the flag is not set, the function will wait until data is available.
TPMC501_FLUSH	If this flag is set the sequencer FIFO will be flushed and the function will wait for new data otherwise the function will read the next available data set.

pData

This argument is a pointer to an array of integer items where the converted data of a sequencer cycle will be filled in. The number of channels and the channel configuration was setup with the tpmc501StartSequencer function. The used buffer must be at least big enough to receive one integer value for every enabled sequencer channel.
The first array item [0] belongs to the channel configured by ChanConfig[0], the second array item [1] belongs to the channel configured by ChanConfig[1] and so forth. Please refer to the example application for details.

pStatus

This argument is a pointer to a variable which returns the actual sequencer error status. Keep in mind to check this status before each reading. If status is 0 no error is pending. A set of bits specifies the error condition:

Value	Description
TPMC501_BUF_OVERRUN	This bit indicates a ring buffer overrun. The error occurred if there is no space in ring buffer to write the new AD data. In this case the new AD values are dismissed. The sequencer was not stopped.
TPMC501_DATA_OVERFLOW	This indicates an overrun in the sequencer data RAM. The error occurred if the driver is too slow to read the data in time. The sequencer was stopped after this error occurred.
TPMC501_TIMER_ERR	Sequencer timer error (see also TPMC501 hardware manual). The sequencer was stopped after this error occurred.
TPMC501_INST_RAM_ERR	Sequencer instruction RAM error (see also TPMC501 hardware manual). The sequencer was stopped after this error occurred.

EXAMPLE

```
#include “tpmc501api.h”

TPMC501_HANDLE   hdl;
TPMC501_STATUS   result;
unsigned int     seqStatus;
int              numOfSeqChannels;
int              *pData;

numOfSeqChannels = 2;        /* Two channels used in sequencer mode */

/* allocate sequence input buffer */
pData = malloc(sizeof(int) * numOfSeqChannels);

…
```

```
…

/* read a set of fresh ADC data */
result = tpmc501GetDataBuffer(hdl, TPMC501_FLUSH, pData, &seqStatus);
if (result != TPMC501_OK)
{
     /* handle error */
}
…
free(pData);
```

RETURN VALUE

On success, TPMC501_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC501_ERR_INVALID_HANDLE	The specified TPMC501_HANDLE is invalid.
TPMC501_ERR_TIMEOUT	The expected wait time has been exceeded.
TPMC501_ERR_NODATA	The function returned without data
TPMC501_ERR_BUSY	The device is not configured in sequencer mode

4 Appendix

4.1 Example Applications

The example application shall give an overview about the use of the TPMC501 devices and how to use the TPMC501 API.

The example application is designed as an interactive console application, so make sure to properly redirect the standard input and standard output for the example application's address space. If using a Dynamic Download Build e.g. in a telnet shell, use the following command:

```
# run -filtered <example_filename> -args <example_address_space>
```