The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC500-SW-42

VxWorks Device Driver

32 Channel 12-bit ADC PMC

Version 6.1.x

User Manual

Issue 6.1.0

May 2022

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

TPMC500-SW-42

VxWorks Device Driver

32 Channel 12-bit ADC PMC

Supported Modules:
TPMC500

Issue	Description	Date
1.0	First Issue	November, 1999
1.1	General Revision	November 24, 2003
2.0.0	New Device Initialization Parameters, File list modified, Support for Intel x86 added	October 29, 2004
3.0.0	New driver startup functions, File list changed, new prefix for defines and structures, read() function replaced by a new ioctl() function	October 1, 2007
4.0.0	Implementation of VxBus Driver, tpmc500init() function added, API added	September 14, 2010
5.0.0	64bit support added, parameter types changes, general revision	December 21, 2011
6.0.0	API function modified, VxWorks 7 support added, new installation guide	July 5, 2018
6.1.0	ioctl for RTP-Support modified	May 20, 2022

Table of Contents

1 Introduction

The TPMC500-SW-42 VxWorks device driver software allows the operation of the supported PMC conforming to the VxWorks I/O system specification.

The TPMC500-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new 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.

Both drivers, legacy and VxBus, share the same application programming interface (API).

Both drivers invoke a mutual exclusion and binary semaphore mechanism to prevent simultaneous requests by multiple tasks from interfering with each other.

The TPMC500-SW-42 device driver supports the following features:

- start AD conversion and read data
- choosing gain, channel, input interface
- 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 TPMC500-SW-42 supports the modules listed below:

TPMC500	32(16) Channel - 12-bit ADC	(PMC)

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

TPMC500 User Manual
TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide

2 API Documentation

2.1 General Functions

2.1.1 tpmc500Open

NAME

tpmc500Open() – opens a device.

SYNOPSIS

```
TPMC500_HANDLE tpmc500Open
(
      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 TPMC500 device is named “/tpmc500/0”, the second device is named “/tpmc500/1” and so on.

EXAMPLE

```
#include “tpmc500.h”

TPMC500_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tpmc500Open(“/tpmc500/0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device descriptor pointer, 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 tpmc500Close

NAME

tpmc500Close() – closes a device.

SYNOPSIS

```
TPMC500_STATUS tpmc500Close
(
     TPMC500_HANDLE       hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

This value specifies the file descriptor pointer to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include “tpmc500.h”

TPMC500_HANDLE     hdl;
TPMC500_STATUS     result;

/*
** close file descriptor to device
*/
result = tpmc500Close(hdl);
if (result != TPMC500_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.3 tpmc500GetModuleInfo

NAME

tpmc500GetModuleInfo – Get module information data

SYNOPSIS

```
TPMC500_STATUS tpmc500GetModuleInfo
(
    TPMC500_HANDLE          hdl,
    TPMC500_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];
} TPMC500_INFO_BUFFER;
```

Variant

This parameter returns the configured module variant (e.g. 10 for a TPMC500-10).

PciBusNo, PciDevNo

These parameters specifies the PCI location of this module

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 “tpmc500api.h”

TPMC500_HANDLE        hdl;
TPMC500_STATUS        result;
TPMC500_INFO_BUFFER   ModuleInfo

result = tpmc500GetModuleInfo(hdl, &ModuleInfo);
if (result != TPMC500_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.

2.2 Device Access Functions

2.2.1 tpmc500SetModelType

NAME

tpmc500SetModelType – configures the TPMC500 board type

SYNOPSIS

```
TPMC500_STATUS tpmc500SetModelType
(
        TPMC500_HANDLE      hdl,
        int                 ModuleType
)
```

DESCRIPTION

This function configures the TPMC500 board type.

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

PARAMETERS

hdl

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

ModlueType

This parameter specifies the model type of the TPMC500. The following model types are supported.

Model Type	Description
10	TPMC500-10 --- input range +/- 10V, gains: 1,2,5,10, front panel I/O
11	TPMC500-11 --- input range +/- 10V, gains: 1,2,4,8, front panel I/O
12	TPMC500-12 --- input range 0…10V, gains: 1,2,5,10, front panel I/O
13	TPMC500-13 --- input range 0…10V, gains: 1,2,4,8, front panel I/O
20	TPMC500-20 --- input range +/- 10V, gains: 1,2,5,10, back I/O
21	TPMC500-21 --- input range +/- 10V, gains: 1,2,4,8, back I/O
22	TPMC500-22 --- input range 0…10V, gains: 1,2,5,10, back I/O
23	TPMC500-23 --- input range 0…10V, gains: 1,2,4,8, back I/O

EXAMPLE

```
#include “tpmc500.h”

TPMC500_HANDLE     hdl;
TPMC500_STATUS     result;

/*
** tell the driver, this is a TPMC500-10
*/
result = tpmc500SetModelType(hdl, 10);
if (result != TPMC500_OK)
{
     /* handle error */
}
```

RETURN VALUE

OK if function succeeds or ERROR.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVAL	A NULL pointer is referenced for an input value
TPMC500_ERR_INVALID_HANDLE	The device handle is invalid
TPMC500_ERR_CONFIG	Unsupported or invalid TPMC500 model type specified
TPMC500_ERR_BUSY	The module is busy

2.2.2 tpmc500Read

NAME

tpmc500Read – perform AD conversion and read value

SYNOPSIS

```
TPMC500_STATUS tpmc500Read
(
    TPMC500_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 (TPMC500_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
TPMC500_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC500_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC500 EEPROM.
TPMC500_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 “tpmc500api”

TPMC500_HANDLE     hdl;
TPMC500_STATUS     result;
int                in_value;

/*
** read AD value from channel 5 with gain = 2
** single endend input, correction enabled, use interrupts
*/
result = tpmc500Read(hdl,
                     5,
                     2,
                     TPMC500_CORR | TPMC500_FAST,
                     &in_value);
if (result != TPMC500_OK)
{
     /* handle error */
}
else
{
     printf(“ADC #5: %d\n”, in_value);
}
```

RETURN VALUE

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC500_ERR_BUSY	The device is busy
TPMC500_ERR_INVAL	Invalid parameter specified: invalid channel number, gain or flag specified
TPMC500_ERR_TIMEOUT	The conversion timed out
TPMC500_ERR_CONFIG	TPMC500 model type unknown or not configured

2.2.3 tpmc500StartSequencer

NAME

tpmc500StartSequencer – setup and start sequencer operation

SYNOPSIS

```
TPMC500_STATUS tpmc500StartSequencer
(
    TPMC500_HANDLE      hdl,
    unsigned int        CycleTime,
    unsigned int        NumOfBufferPages,
    unsigned int        NumOfChannels,
    TPMC500_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. Additional the sequencer cycle time is defined and depth of the drivers 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 100 µs 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 100 µs to 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;
} TPMC500_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 *TPMC500-10/-12/-20/-22* and 1, 2, 4, 8 for *TPMC500-11/-13/-21/-23.*

flags

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

Flag	Meaning
TPMC500_DIFF	If this flag is set the ADC input works in differential mode otherwise in single-ended (default).
TPMC500_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC500 EEPROM.

EXAMPLE

```
#include "tpmc500api.h"

TPMC500_HANDLE     hdl;
TPMC500_STATUS     result;
unsigned int       CycleTime;
unsigned int       NumOfBufferPages;
unsigned int       NumOfChannels;
TPMC500_CHAN_CONF  ChanConf[TPMC500_MAX_CHAN];

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

…
```

```
…

ChanConf[0].ChanToUse   = 1;
ChanConf[0].gain        = 1;
ChanConf[0].flags       = TPMC500_CORR;

ChanConf[1].ChanToUse   = 20;
ChanConf[1].gain        = 5;
ChanConf[1].flags       = TPMC500_CORR;
…

// start the sequencer
result = tpmc500StartSequencer(hdl,
                               CycleTime,
                               NumOfBufferPages,
                               NumOfChannels,
                               ChanConf);
if (result != TPMC500_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC500_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.
TPMC500_ERR_INVAL	At least one of the parameters is invalid.
TPMC500_ERR_CONFIG	TPMC500 model type unknown or not configured
TPMC500_ERR_NOMEM	Allocating buffer failed

2.2.4 tpmc500StopSequencer

NAME

tpmc500StopSequencer – Stop the sequencer

SYNOPSIS

```
TPMC500_STATUS tpmc500StopSequencer
(
     TPMC500_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 “tpmc500api.h”

TPMC500_HANDLE   hdl;
TPMC500_STATUS   result;

result = tpmc500StopSequencer(hdl);
if (result != TPMC500_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified device handle is invalid

2.2.5 tpmc500GetDataBuffer

NAME

tpmc500GetDataBuffer – Get next data block of sequencer samples

SYNOPSIS

```
TPMC500_STATUS tpmc500GetDataBuffer
(
        TPMC500_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, although 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
TPMC500_NOWAIT	If this flag is set the function will return immediately, although there is no data available. If the flag is not set, the function will wait until data is available.
TPMC500_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 tpmc500StartSequencer function. The used buffer must be at least big enough to receive on 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
TPMC500_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.
TPMC500_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.
TPMC500_TIMER_ERR	Sequencer timer error (see also TPMC500 hardware manual). The sequencer was stopped after this error occurred.
TPMC500_INST_RAM_ERR	Sequencer instruction RAM error (see also TPMC500 hardware manual). The sequencer was stopped after this error occurred.

EXAMPLE

```
#include “tpmc500api.h”

TPMC500_HANDLE   hdl;
TPMC500_STATUS   result;
unsigned int     seqStatus;
int              numOfSeqChannels;
int              *pData;

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

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

…
```

…

```
/* read a set of fresh ADC data */
result = tpmc500GetDataBuffer(hdl, TPMC500_FLUSH, &pData, &seqStatus);
if (result != TPMC500_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC500_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC500_ERR_INVALID_HANDLE	The specified TPMC500_HANDLE is invalid.
TPMC500_ERR_TIMEOUT	There the expected wait time has been exceeded.
TPMC500_ERR_NOT_READY	The sequencer is stopped.
TPMC500_ERR_NODATA	The function returned without data
TPMC500_ERR_BUSY	The device is not configured in sequencer mode

3 Legacy I/O System Functions

This chapter describes the legacy driver-level interface to the I/O system. The purpose of these functions is to install the driver in the I/O system, add and initialize devices.

The legacy I/O system functions are only relevant for the legacy TPMC500 driver. For the VxBus-enabled TPMC500 driver, the driver will be installed automatically in the I/O system and devices will be created as needed for detected modules.

3.1 tpmc500PciInit

NAME

tpmc500PciInit() – Generic PCI device initialization

SYNOPSIS

void tpmc500PciInit()

DESCRIPTION

This function is required only for Intel x86 VxWorks platforms. The purpose is to setup the MMU mapping for all required TPMC500 PCI spaces (base address register) and to enable the TPMC500 device for access.

The global variable *tpmc500Status* obtains the result of the device initialization and can be polled later by the application before the driver will be installed.

Value	Meaning
> 0	Initialization successful completed. The value of tpmc500Status is equal to the number of mapped PCI spaces
0	No TPMC500 device found
< 0	Initialization failed. The value of (tpmc500Status & 0xFF) is equal to the number of mapped spaces until the error occurs. Possible cause: Too few entries for dynamic mappings in sysPhysMemDesc[]. Remedy: Add dummy entries as necessary (syslib.c).

EXAMPLE

```
extern void tpmc500PciInit();

…

tpmc500PciInit();
```

4 Appendix

4.1 Enable RTP-Support

Using TPMC500 devices tunneled from Real Time Processes (RTPs) is implemented. For this the “TEWS TPMC500 IOCTL command validation” must be enabled in system configuration.

The API source file “tpmc500api.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”.

All legacy functions, functions for version compatibility and debugging functions are not usable from RTPs.

4.2 Debugging and Diagnostic

The TPMC500 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 TPMC500 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 rare the function can be removed from the code by un-defining the macro INCLUDE_TPMC500_SHOW in tpmc500drv.c

The tpmc500Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TPMC500 modules and device statistics.

If TPMC500 modules were probed but no devices were created it may helpful to enable debugging code inside the driver code by defining the macro TPMC500_DEBUG in tpmc500drv.c.

In contrast to VxBus TPMC500 devices, legacy TPMC500 devices must be created “manually”. This will be done with the first call to the tpmc500Open API function.

```
-> tpmc500Show
Probed Modules:
    [0] TPMC500: Bus=4, Dev=1, DevId=0x9050, VenId=0x10b5, Init=OK, vxDev=0x478878

Associated Devices:
    [0] TPMC500: /tpmc500/0

Correction Data:
    /tpmc500/0:
                      gain/offset
        Gain = 1:      5/-8
        Gain = 2:      3/-2
```

```
            Gain = 4/5:   11/-5
            Gain = 8/10:  7/-1

    Device Statistics:
        /tpmc500/0:
            open count = 0
            interrupt count = 3
```