The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC600-SW-42

VxWorks Device Driver

32 (16) Digital Inputs

Version 4.0.x

User Manual

Issue 4.0.0

September 2025

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

TPMC600-SW-42

VxWorks Device Driver

32 (16) Digital Inputs

Supported Modules:
TPMC600

Issue	Description	Date
1.0	First Issue	May 1, 2001
1.1	General Revision	November 2003
1.1.1	File-list changed	August 10, 2005
2.0.0	Functions tpmc600Drv(), tpmc600DevCreate modified Filelist changed	March 9, 2007
3.0.0	VxBus-Support added, driver API added “Application dependent adjustments” removed read() replaced by ioctl()-function FIO_TPMC600_READ	September 20, 2010
4.0.0	Address of TEWS Technologies changed Support for Legacy VxWorks (before V 6.9) is EOL, descriptions removed Description of Basic I/O access removed New function tpmc600GetModuleInfo() added General API update	September 12, 2025

Table of Contents

1 Introduction

1.1 Device Driver

The TPMC600-SW-42 VxWorks device driver software allows the operation of the supported PMC conforming to the VxWorks I/O system specification.

The TPMC600-SW-42 release contains driver sources for the VxBus-enabled (GEN1 and GEN2) driver model and supports later VxWorks 6.9.x and VxWorks 7 releases, including VxWorks 64-bit and SMP systems.

The driver provides an application programming interface (API) for easy access to all functionalities.

The driver invokes synchronization methods to prevent simultaneous requests by multiple tasks from interfering with each other.

The TPMC600-SW-42 device driver supports the following features:

- Read the current input value
- Wait for selectable input events (match, high-, low-, any transition on the input line(s))
- Configure, start and stop input debouncing
- Read board information

The TPMC600-SW-42 supports the modules listed below:

TPMC600-x0	32 Digital Inputs	(PMC)
TPMC600-x1	16 Digital Inputs	(PMC)

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

TPMC600 User Manual
TEWS Technologies VxWorks Device Drivers - Installation Guide

2 API Documentation

2.1 General Functions

2.1.1 tpmc600Open

NAME

tpmc600Open – opens a device

SYNOPSIS

```
TPMC600_HANDLE tpmc600Open
(
      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 TPMC600 device is named “/tpmc600/0”, the second device is named “/tpmc600/1” and so on.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;

/* open the specified device */
hdl = tpmc600Open(“/tpmc600/0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

The error codes are stored in *errno.*

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

2.1.2 tpmc600Close

NAME

tpmc600Close – closes a device

SYNOPSIS

```
TPMC600_STATUS tpmc600Close
(
     TPMC600_HANDLE       hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

pDev

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

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;

/*
** close file descriptor to device
*/
result = tpmc600Close(hdl);
if (result != TPMC600_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.3 tpmc600GetModuleInfo

NAME

Tpmc600GetModuleInfo – Get module information data

SYNOPSIS

```
TPMC600_STATUS tpmc600GetModuleInfo
(
    TPMC600_HANDLE          hdl,
    TPMC600_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;
} TPMC600_INFO_BUFFER;
```

Variant

This parameter returns the configured module variant (e.g. 10 for a TPMC600-10).

PciBusNo, PciDevNo

These parameters specifies the PCI location of this module

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE        hdl;
TPMC600_STATUS        result;
TPMC600_INFO_BUFFER   moduleInfo

result = tpmc600GetModuleInfo(hdl, &moduleInfo);
if (result != TPMC600_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The specified TPMC600_HANDLE is invalid.

2.2 Device Access Functions

2.2.1 tpmc600Read

NAME

tpmc600Read – read input state of device

SYNOPSIS

```
TPMC600_STATUS tpmc600Read
(
    TPMC600_HANDLE      hdl,
    unsigned int        *pDigInVal
)
```

DESCRIPTION

This function reads the current input state of the device.

PARAMETERS

hdl

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

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant supporting less than 32 input lines is used, the unused bits will be set to 0.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;
unsigned int       in_value;

/* read current state of I/O lines */
result = tpmc600Read(hdl, &in_value);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     printf(“input value: 0x%08X\n”, in_value);
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	A NULL pointer is referenced for an input value
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.2 tpmc600EnableDebouncer

NAME

tpmc600EnableDebouncer – configure and enable input debouncer

SYNOPSIS

```
TPMC600_STATUS tpmc600EnableDebouncer
(
    TPMC600_HANDLE      hdl,
    unsigned short      debValue
)
```

DESCRIPTION

This function configures the input debouncer, which shall prevent detecting fast faulty signal changes.

PARAMETERS

hdl

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

debValue

This argument specifies the debouncer timer value. Valid values are between 0 for 7us and 65535 for 440ms. Please refer to the TPMC600 User Manual for a detailed description.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;

/* enable debouncer with a debounce time of ~1ms (143) */
result = tpmc600EnableDebouncer(hdl, 143);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* function successfully completed */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.3 tpmc600DisableDebouncer

NAME

tpmc600DisableDebouncer – disable input debouncer

SYNOPSIS

```
TPMC600_STATUS tpmc600DisableDebouncer
(
     TPMC600_HANDLE          hdl
)
```

DESCRIPTION

This function disables the input debouncer.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;

/* disable debouncer */
result = tpmc600DisableDebouncer(hdl);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* function successfully completed */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.4 tpmc600WaitForAnyEvent

NAME

tpmc600WaitForAnyEvent – wait for transition on input line

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForAnyEvent
(
    TPMC600_HANDLE      hdl,
    int                 inputLine,
    int                 msTimeout
)
```

DESCRIPTION

This function waits for any transition on the specified input line.

PARAMETERS

hdl

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

inputLine

This argument specifies the input line. A value of 1 must be specified for IN1, 2 for IN2 and so on. The maximum valid input line depends on the used module variant.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;

/* wait for a transition (any) on IN12
**     timeout after approximal 10 sec. */
result = tpmc600WaitForAnyEvent(hdl, 12, 10000);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* event occurred */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	An invalid parameter value (inputLine) has been specified
TPMC600_ERR_NOMEM	Waiting job can’t be created, memory allocation failed
TPMC600_ERR_TIMEOUT	The waiting job timed out
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.5 tpmc600WaitForHighEvent

NAME

tpmc600WaitForHighEvent – wait for low-to-high transition on input line

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForHighEvent
(
    TPMC600_HANDLE      hdl,
    int                 inputLine,
    int                 msTimeout
)
```

DESCRIPTION

This function waits for a low-to-high transition on the specified input line.

PARAMETERS

hdl

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

inputLine

This argument specifies the input line. A value of 1 must be specified for IN1, 2 for IN2 and so on. The maximum valid input line depends on the used module variant.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;

/* wait for a low-to-high transition on IN7
**     timeout after approximal 0.5 sec. */
result = tpmc600WaitForHighEvent(hdl, 7, 500);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* event occurred */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	An invalid parameter value (inputLine) has been specified
TPMC600_ERR_NOMEM	Waiting job can’t be created, memory allocation failed
TPMC600_ERR_TIMEOUT	The waiting job timed out
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.6 tpmc600WaitForLowEvent

NAME

tpmc600WaitForLowEvent – wait for a high-to-low transition on input line

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForLowEvent
(
    TPMC600_HANDLE      hdl,
    int                 inputLine,
    int                 msTimeout
)
```

DESCRIPTION

This function waits for a high-to-low transition on the specified input line.

PARAMETERS

hdl

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

inputLine

This argument specifies the input line. A value of 1 must be specified for IN1, 2 for IN2 and so on. The maximum valid input line depends on the used module variant.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;

/* wait for a high-to-low transition on IN5
**     never timeout */
result = tpmc600WaitForLowEvent(hdl, 5, -1);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* event occurred */
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	An invalid parameter value (inputLine) has been specified
TPMC600_ERR_NOMEM	Waiting job can’t be created, memory allocation failed
TPMC600_ERR_TIMEOUT	The waiting job timed out
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.7 tpmc600WaitForMultiAnyEvents

NAME

tpmc600WaitForMultiAnyEvents – wait for 1st transition on set of input lines and return input state

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForMultiAnyEvents
(
    TPMC600_HANDLE      hdl,
    unsigned int        lineMask,
    int                 msTimeout,
    unsigned int        *pDigInVal
)
```

DESCRIPTION

This function waits for the first transition on any of the specified input lines. After detection of the transition the input state will be read and returned.

There is a delay between the transition and actual reading of the input state. This means that the returned value represents the input state a short time after the transition has occurred.

PARAMETERS

hdl

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

lineMask

This argument specifies a mask of input lines that are active to wait for a transition. A set bit specifies an active input line. Input lines set to 0 will not be observed. Bit 0 is assigned to IN1, bit 1 to IN2, and so on.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;
unsigned int       inVal;

/* read input state after 1st transition (any)
**     on IN1, IN2, IN3, IN4, or IN16
**     timeout after approx. 10 sec. */
result = tpmc600WaitForMultiAnyEvents(hdl,
                                      0x0000800F,
                                      10000,
                                      (int)&inVal);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* event occurred */
     printf(“Input Value after event: %08Xh\n”, inVal);
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	A NULL pointer is referenced for an input value
TPMC600_ERR_NOMEM	Waiting job can’t be created, memory allocation failed
TPMC600_ERR_TIMEOUT	The waiting job timed out
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.8 tpmc600WaitForMultiHighEvents

NAME

tpmc600WaitForMultiHighEvents – wait for 1st low-to-high transition on set of input lines and return input state

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForMultiHighEvents
(
    TPMC600_HANDLE      hdl,
    unsigned int        lineMask,
    int                 msTimeout,
    unsigned int        *pDigInVal
)
```

DESCRIPTION

This function waits for the first low-to-high transition on any of the specified input lines. After detection of the transition the input state will be read and returned.

There is a delay between the transition and actual reading of the input state. This means that the returned value represents the input state a short time after the transition has occurred.

PARAMETERS

hdl

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

lineMask

This argument specifies a mask of input lines that are active to wait for a transition. A set bit specifies an active input line, input lines set to 0 will not be observed. Bit 0 is assigned to IN1, bit 1 to IN2, and so on.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;
unsigned int       inVal;

/* read input state after 1st low-to-high transition
**     on IN2, or IN16
**     timeout after approximal 1 sec. */
result = tpmc600WaitForMultiHighEvents(hdl,
                                       0x00008002,
                                       1000,
                                       (int)&inVal);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* high event occurred */
     printf(“Input Value after event: %08Xh\n”, inVal);
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	A NULL pointer is referenced for an input value
TPMC600_ERR_NOMEM	Waiting job can’t be created, memory allocation failed
TPMC600_ERR_TIMEOUT	The waiting job timed out
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

2.2.9 tpmc600WaitForMultiLowEvents

NAME

tpmc600WaitForMultiLowEvents – wait for 1st high-to-low transition on set of input lines and return input state

SYNOPSIS

```
TPMC600_STATUS tpmc600WaitForMultiLowEvents
(
    TPMC600_HANDLE      hdl,
    unsigned int        lineMask,
    int                 msTimeout,
    unsigned int        *pDigInVal
)
```

DESCRIPTION

This function waits for the first high-to-low transition on any of the specified input lines. After detection of the transition the input state will be read and returned.

There is a delay between the transition and actual reading of the input state. This means that the returned value represents the input state a short time after the transition has occurred.

PARAMETERS

hdl

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

lineMask

This argument specifies a mask of input lines that are active to wait for a transition. A set bit specifies an active input line, input lines set to 0 will not be observed. Bit 0 is assigned to IN1, bit 1 to IN2, and so on.

msTimeout

This argument specifies the maximum time in milliseconds the function will wait for the specified event. If the time elapses without the event occurred, the function will return with an adequate error. A value of -1 specifies that the function never times out.

pDigInVal

This argument points to a buffer where the state of the input lines will be returned. Bit 0 of the returned value represents the state of IN1, bit 1 of IN2, and so on. If a module variant is used, supporting less than 32 input lines, the unused bits will be set to 0.

EXAMPLE

```
#include “tpmc600api.h”

TPMC600_HANDLE     hdl;
TPMC600_STATUS     result;
unsigned int       inVal;

/* read input state after a high-to-low transition
**     on any input line
**     never timeout */
result = tpmc600WaitForMultiLowEvents(hdl,
                                      0xFFFFFFFF,
                                      -1,
                                      (int)&inVal);
if (result != TPMC600_OK)
{
     /* handle error */
}
else
{
     /* high-to-low event occurred */
     printf(“Input Value after event: %08Xh\n”, inVal);
}
```

RETURNS

On success, TPMC600_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC600_ERR_INVAL	A NULL pointer is referenced for an input value
TPMC600_ERR_NOMEM	Waiting job can’t be created, memory allocation failed
TPMC600_ERR_TIMEOUT	The waiting job timed out
TPMC600_ERR_INVALID_HANDLE	The device handle is invalid

3 Appendix

3.1 Enable RTP-Support

Using TPMC600 devices tunneled from Real Time Processes (RTPs) is implemented. For this the “TEWS TPMC600 IOCTL command validation” must be enabled in system configuration.

The API source file “tpmc600api.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”.

Debugging functions are not usable from RTPs.

3.2 Debugging and Diagnostic

The TPMC600 device driver provides a function and debug statements to display versatile information of the driver installation and status on the debugging console.

The TPMC600 show routine is included in the VxBus 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_TPMC600_SHOW in tpmc600drv.c

The tpmc600Show function displays detailed information about probed modules, assignment of devices respective device names to probed TPMC600 modules and device statistics.

If TPMC600 modules were probed but no devices were created it may be helpful to enable debugging code inside the driver code by defining the macro TPMC600_DEBUG in tpmc600drv.c.

```
-> tpmc600Show
Probed Modules:
    [0] TPMC600-10: Bus=4, Dev=1, DevId=0x0258, VenId=0x1498, Init=OK, vxDev=0x140710

Associated Devices:
    [0] TPMC600-10: /tpmc600/0

Device Configuration:
    /tpmc600/0:
        debouncer = disabled
        pending event wait jobs = 0

Device Statistics:
    /tpmc600/0:
        open count = 0
        interrupt count = 0
        handled events = 0
```