The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC551-SW-42

VxWorks Device Driver

8/4 Channel 16 Bit D/A

Version 4.1.x

User Manual

Issue 4.1.0

August 2024

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

TPMC551-SW-42

VxWorks Device Driver

8/4 Channel 16 Bit D/A

Supported Modules:
TPMC551

Issue	Description	Date
1.0	First Issue	July 15, 1999
1.1	Support for Intel x86 based targets	June 19, 2000
1.2	General Revision	November 28, 2003
1.3.0	File list changed (Release.txt added)	August 10, 2005
2.0.0	New Parameter Interfaces tpmc551Drv(), tpmc551DevCreate() File list changed (ChangeLog.txt added)	March 7, 2007
3.0.0	API description added, VxBus support added, file list modified I/O interface modified, address TEWS LLC removed	February 1, 2011
4.0.0	API Interface reviewed, General Revision of the Manual	May 27, 2013
4.1.0	VxWorks 7 support added. Installation moved to a separate manual. New Address of TEWS Technologies GmbH	August 2, 2024

Table of Contents

1 Introduction

The TPMC551-SW-42 VxWorks device driver software allows the operation of the supported PMC conforming to the VxWorks I/O system specification.

The TPMC551-SW-42-SW-42 release contains independent driver sources for the old legacy (pre-VxBus) and the new VxBus-enabled driver model. The VxBus-enabled driver is recommended for new developments with later VxWorks 6.x release and mandatory for VxWorks SMP systems.

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

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

The TPMC551-SW-42 device driver supports the following features:

- Setting DAC output value
- Configure, start, and stop DAC-sequencer
- Write data for sequencer cycle
- Use of data correction for simple conversion and in sequencer mode
- Use of latched writes for synchronous output
- Reading TPMC551 configuration (number of channels and uni-/bipolar output)

The TPMC551-SW-42 supports the modules listed below:

TPMC551-10	8 channel 16-bit D/A (Front I/O)	(PMC)
TPMC551-11	4 channel 16-bit D/A (Front I/O)	(PMC)
TPMC551-21	8 channel 16-bit D/A (Back I/O)	(PMC)
TPMC551-21	4 channel 16-bit D/A (Back I/O)	(PMC)

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

TEWS TECHNOLOGIES VxWorks Device Drivers - Installation Guide
TPMC551 User Manual

2 API Documentation

2.1 General Functions

2.1.1 tpmc551Open

NAME

tpmc551Open – opens a device.

SYNOPSIS

```
TPMC551_HANDLE tpmc551Open
(
        char            *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle must be opened by a call to this function. If the legacy TPMC551 driver is used, this function will also install the legacy driver and create devices with the first call. The VxBus TPMC551 driver will be installed automatically by the VxBus system.

The tpmc551Open function can be called multiple times (e.g. in different tasks).

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	/tpmc551/0
2	/tpmc551/1

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tpmc551Open(“/tpmc551/0”);
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 tpmc551Close

NAME

tpmc551Close – closes a device.

SYNOPSIS

```
TPMC551_STATUS tpmc551Close
(
     TPMC551_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 “tpmc551api.h”

TPMC551_HANDLE     hdl;
TPMC551_STATUS     result;

/*
** close the device
*/
result = tpmc551Close(hdl);
if (result != TPMC551_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified device handle is invalid

2.1.3 tpmc551GetModuleInfo

NAME

tpmc551GetModuleInfo – Get module information

SYNOPSIS

```
TPMC551_STATUS tpmc551GetModuleInfo
(
    TPMC551_HANDLE      hdl,
    int                 *NumChan,
    int                 bipolar[TPMC551_MAX_CHAN],
    int                 OffsCorr[TPMC551_MAX_CHAN],
    int                 GainCorr[TPMC551_MAX_CHAN]
)
```

DESCRIPTION

This function reads module information data from the specified device.

PARAMETERS

hdl

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

NumChan

This argument is a pointer to an int variable where the number of available DAC channels is returned.

bipolar

This argument is a pointer to an int array where the configured voltage range of each DAC channel is returned as boolean value. The array element bipolar[0] contains the range stetting for DAC channel 1, bipolar[1] for DAC channel 2 and so forth. If the corresponding value is TRUE then the voltage range of the channel is configured to +/- 10V output (bipolar); otherwise it is configured to 0…10V output voltage range.

OffsCorr

This argument is a pointer to an int array where the factory programmed offset correction data is returned. OffsCorr[0] contains correction data for DAC channel 1, OffsCorr[1] for DAC channel 2 and so forth.

GainCorr

This argument is a pointer to an int array where the factory programmed gain correction data are returned. GainCorr[0] contains correction data for DAC channel 1, GainCorr[1] for DAC channel 2 and so forth.

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE   hdl;
TPMC551_STATUS   result;
int              NumChan;
int              bipolar[TPMC551_MAX_CHAN];
int              OffsCorr[TPMC551_MAX_CHAN];
int              GainCorr[TPMC551_MAX_CHAN];

/* Get module information data */

result = tpmc551GetModuleInfo(hdl, &NumChan, bipolar, OffsCorr, GainCorr);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.

2.2 DAC Output Functions

2.2.1 tpmc551DacWrite

NAME

tpmc551DacWrite – write D/A value to specified channel

SYNOPSIS

```
TPMC551_STATUS tpmc551DacWrite
(
    TPMC551_HANDLE      hdl,
    int                 channel,
    unsigned int        flags,
    int                 value
)
```

DESCRIPTION

This function writes a new value to a specific channel and starts D/A conversion immediately in transparent mode

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 DAC channel which shall be updated. Possible values are 1 up to the number of available DAC channels of the specific module.

flags

This argument specifies a set of bit flags that control the D/A conversion:

Value	Description
TPMC551_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC551 EEPROM.

value

This argument specifies the new 16-bit D/A value. Valid data range depends on the voltage range of the specified channel (0…65535 for 0...10V voltage range and -32768…32767 for +/-10V voltage range).

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE   hdl;
TPMC551_STATUS   result;

result = tpmc551DacWrite(hdl, 1, TPMC551_CORR, 12345);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.
TPMC551_ERR_RANGE	Invalid channel number
TPMC551_ERR_TIMEOUT	Timeout during D/A conversion
TPMC551_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

2.2.2 tpmc551DacWriteMulti

NAME

tpmc551DacWriteMulti – write D/A value to multiple channels

SYNOPSIS

```
TPMC551_STATUS tpmc551DacWriteMulti
(
    TPMC551_HANDLE      hdl,
    unsigned int        ChannelMask,
    unsigned int        flags,
    int                 values[TPMC551_MAX_CHAN]
)
```

DESCRIPTION

This function writes new values to specified channels and starts D/A conversion immediately (transparent mode) or simultaneously (latched mode).

PARAMETERS

hdl

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

ChannelMask

This argument selects DAC channels which shall be updated. A set (1) bit specifies that the corresponding channel shall be updated. Bit 0 corresponds to the first DAC channel, bit 1 corresponds to the second DAC channel and so on.

flags

This argument specifies a set of bit flags that control the D/A conversion:

Value	Description
TPMC551_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC551 EEPROM for all selected channels.
TPMC551_SIMCONV	Start conversion of selected channels in latched mode and update analog outputs simultaneously.

values

This array contains the new 16-bit D/A values. Valid data range depends on the voltage range of the specified channel (0…65535 for 0...10V voltage range and -32768…32767 for +/-10V voltage range).
Array index 0 corresponds to the first DAC channel, array index 1 corresponds to the second DAC channel and so on. Only channels selected for update (*ChannelMask)* will be modified.

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE  hdl;
TPMC551_STATUS  result;
unsigned int    ChannelMask;
unsigned int    flags;
int             values[TPMC551_MAX_CHAN];

/* Update channel 1, 4 and 8 simultaneously with corrected D/A values */
ChannelMask = (1<<0) | (1<<3) | (1<<7);
flags = TPMC551_CORR | TPMC551_SIMCONV;
value[0] = 1111;   /* channel 1 */
value[3] = 4444;   /* channel 4 */
value[7] = 8888;   /* channel 8 */

result = tpmc551DacWriteMulti(hdl, ChannelMask, flags, values);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.
TPMC551_ERR_RANGE	Invalid channel number
TPMC551_ERR_TIMEOUT	Timeout during D/A conversion
TPMC551_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

2.3 Sequencer Functions

2.3.1 tpmc551SeqSetup

NAME

tpmc551SeqSetup – Setup sequencer facility

SYNOPSIS

```
TPMC551_STATUS tpmc551SeqSetup
(
    TPMC551_HANDLE      hdl,
    int                 CycleTime,
    int                 NumActiveChannels,
    int                 NumBufTuples,
    int                 ChannelAllocation[TPMC551_MAX_CHAN],
    unsigned int        flags
)
```

DESCRIPTION

This function configures the sequencer facility and allocates memory for the sequencer software ring buffer. The behaviour of the sequencer facility is controlled by a set of bit flags which are described below.

Basically the sequencer will perform a D/A conversion on active channels in a deterministic time period controlled by a cycle timer or the duration of the conversion itself. To be sure that D/A data will be available for the next cycle just in-time, data for the sequencer will be provided by a configurable ring buffer. The ring buffer can be asynchronously filled by the application program.

The sequencer facility provides two operating modes. In loop mode (TPMC551_LOOP) the buffer will be filled completely with new data (e.g. wave form). The contents of the buffer will be output continuously in a loop. In normal mode (TPMC551_LOOP is not set) the application program must provide new data for every cycle. If the buffer is empty then the sequencer will stop and it holds the last output value until new data arrives.

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 sequencer cycle time in steps of 100 µs. This argument is only relevant if the flag TPMC551_TIMERMODE is set.

NumActiveChannels

This argument specifies the number of active channels. Valid range is 1 up to the number of available channels (4 or 8).

NumBufTuples

This argument specifies the size of the sequencer software ring buffer. In this case size is not the number of bytes to allocate but rather the number of tuples (data for all active channels per cycle).

ChannelAllocation

This argument specifies the channel number of active channels and their enumeration inside a tuple. The function tpmc551SeqWrite awaits new data for active channels in this order. The first array element contains the channel number (1...n) of the first active channel. The second array element the channel number of the second active channel and so forth. Unused array elements can be left undefined.

flags

This argument specifies a set of bit flags that control the sequencer operation:

Value	Description
TPMC551_TIMERMODE	If set, the cycle of D/A conversions will be controlled by the sequencer timer in steps of 100 microseconds; otherwise the sequencer will run in continuous mode as fast as possible (based on the conversion time).
TPMC551_LOOP	If this flag is set (loop mode) the ring buffer never becomes empty. Once completely filled the sequencer will continuously get data out of the buffer for the next conversion. If this flag is not set (normal mode) and the buffer becomes empty then the sequencer will stop and it holds the last output value until new data arrives.
TPMC551_CORR	Perform an offset and gain correction with factory calibration data stored in the TPMC551 EEPROM for all selected channels.
TPMC551_SIMCONV	Start conversion of active channels in latched mode and update analog outputs simultaneously.

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE   hdl;
TPMC551_STATUS   result;
int              ChannelAllocation[TPMC551_MAX_CHAN];
unsigned int     flags;

…
```

…

```
/* Setup the sequencer with 2 active channels (1 and 4) in timer mode */
/* with 1 ms cycle time. The sequencer buffer shall store data tuples */
/* for up to 100 cycles. */

ChannelAllocation[0] = 1;
ChannelAllocation[1] = 4;
flags = TPMC551_TIMERMODE | TPMC551_CORR | TPMC551_SIMCONV;

result = tpmc551SeqSetup(hdl, 10, 2, 100, ChannelAllocation, flags);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.
TPMC551_ERR_RANGE	Invalid channel number or invalid number of channels.
TPMC551_ERR_NOMEM	Unable to allocate memory for the ring buffer.
TPMC551_ERR_BUSY	This error occurs if the sequencer is still running. Please stop the sequencer before executing this function.

2.3.2 tpmc551SeqStart

NAME

tpmc551SeqStart – start sequencer facility

SYNOPSIS

```
TPMC551_STATUS tpmc551SeqStart
(
     TPMC551_HANDLE       hdl
)
```

DESCRIPTION

This function starts the sequencer facility. Before calling this function the sequencer must be setup with tpmc551SeqSetup und the ring buffer must be filled with tpmc551SeqWrite.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE   hdl;
TPMC551_STATUS   result;

/* start the sequencer */
result = tpmc551SeqStart(hdl);

if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.
TPMC551_ERR_NOT_READY	The sequencer facility was not properly configured. Execute the function tpmc551SeqSetup first.
TPMC551_ERR_NODATA	No data is available in the ring buffer to start the sequencer facility. Use the function tpmc551SeqWrite to write at least one data tuple before starting the sequencer.

2.3.3 tpmc551SeqStop

NAME

tpmc551SeqStop – stop the sequencer facility

SYNOPSIS

```
TPMC551_STATUS tpmc551SeqStop
(
     TPMC551_HANDLE       hdl
)
```

DESCRIPTION

This function stops the sequencer facility. All allocated resources (e.g. ring buffer memory) will be freed.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE  hdl;
TPMC551_STATUS  result;

/* stop the sequencer */
result = tpmc551SeqStop(hdl);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.

2.3.4 tpmc551SeqWrite

NAME

tpmc551SeqWrite – write new sequencer data

SYNOPSIS

```
TPMC551_STATUS tpmc551SeqWrite
(
        TPMC551_HANDLE      hdl,
        int                 size,
        int                 *values,
        int                 *WrittenSize
);
```

DESCRIPTION

This function writes new data to the sequencers data buffer. The provided data buffer must always contain new data for all active channels (tuple). The number of tuples per write must be at least one up to “unlimited”. This function will always write as many tuples as possible. If the buffer becomes full the function will return immediately with the error TPMC551_ERR_BUF_FULL. The number of written bytes will be returned in a variable pointed to by WrittenSize.

PARAMETERS

hdl

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

size

This argument specifies the size (in bytes) of the data buffer to write.

values

This argument is a pointer to an array of int variables that contains data for all active channels for at least one sequencer cycle (tuple). Despite of the declaration as simple int pointer this array is treated as a two-dimensional array with variable dimensions. The rows of the array represent the number of tuples and the columns the number of active channels. A declaration of this array will look like this: *data[tuples][channels]*.

WrittenSize

This argument is a pointer to an int variable where the number of written bytes is returned. In case of the error TPMC551_ERR_BUF_FULL this value can be used to adjust the buffer start pointer for subsequent writes.

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE  hdl;
TPMC551_STATUS  result;
int             WrittenSize;
int             ForOneCycle[4];
int             ForHundredCycles[100][4];

/* Fill new data into the data buffers */
ForHundredCycles[0][0] = 1;      /* first cycle, first channel */
ForHundredCycles[0][1] = 2;      /* first cycle, second channel */
...
ForHundredCycles[1][0] = 11;     /* second cycle, first channel */
...
ForHundredCycles[99][3] = 1234;  /* 100th cycle, last channel */

/* Write new data for 100 cycles and 4 active channels (100 * 4 values) */
result = tpmc551SeqWrite(
              hdl,
              sizeof(ForHundredCycles),
              (int*)ForHundredCycles,
              &WrittenSize);
if (result != TPMC551_OK)
{
     /* handle error */
     if (result == TPMC551_ERR_BUF_FULL)
     {
          /* send remaining data later */
     }
}

/* Write new data for 1 cycle and 4 active channels (4 values) */
result = tpmc551SeqWrite(
              hdl,
              sizeof(ForOneCycle),
              ForOneCycle,
              &WrittenSize);
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.
TPMC551_ERR_NOT_READY	The sequencer is not running
TPMC551_ERR_BUF_TOO_SMALL	The buffer does not contain enough data for all active channels.
TPMC551_ERR_NOMEM	The passed data buffer does not fit into the configured sequencer buffer. This error is only relevant in loop mode (TPMC551_LOOP)
TPMC551_ERR_BUF_FULL	The sequencer buffer is full. Not all data was written to the buffer. Use the contents of WrittenSize to adjust the data pointer to write the remaining data tuples.

2.3.5 tpmc551SeqFlush

NAME

tpmc551SeqFlush – flush the sequencer ring buffer

SYNOPSIS

```
TPMC551_STATUS tpmc551SeqFlush
(
     TPMC551_HANDLE       hdl
);
```

DESCRIPTION

This function flushes the ring buffer of the sequencer facility. The analog output of active channels will hold the last converted data until new data is written with the tpmc551SeqWrite function.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE  hdl;
TPMC551_STATUS  result;

/* flush the sequencer ring buffer */
result = tpmc551SeqFlush(hdl);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.

2.3.6 tpmc551SeqStatus

NAME

tpmc551SeqStatus – get sequencer status and statistic information

SYNOPSIS

```
TPMC551_STATUS tpmc551SeqStatus
(
    TPMC551_HANDLE      hdl,
    int                 *OperatingState,
    int                 *status,
    int                 *CycleCount,
    int                 *UnderflowCount,
    int                 *EmptyCount
);
```

DESCRIPTION

This function reads sequencer status and statistic information from the specified device.

PARAMETERS

hdl

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

OperatingState

This argument is a pointer to an int variable where the current operating state of the sequencer is returned. Possible operating states are:

Value	Description
TPMC551_STOPPED	The sequencer is stopped.
TPMC551_READY	The sequencer facility is configured and ready to start.
TPMC551_RUNNING	The sequencer is running.

status

This argument is a pointer to an int variable where current error/status of the sequencer is returned. After calling this function the error/status code will be set to TPMC551_SEQ_OK.
Possible error/status codes are:

Value	Description
TPMC551_SEQ_OK	Sequencer is working fine. No errors detected.
TPMC551_SEQ_UNDERFLOW	The sequencer hardware has detected a data underflow condition. The driver was not able to provide new data within a sequencer timer cycle.
TPMC551_SEQ_NODATA	No data available in the ring buffer for output.

CycleCount

This argument is a pointer to an int variable where the total number of sequencer cycles since sequencer start is returned.

UnderflowCount

This argument is a pointer to an int variable where the total number of sequencer underflows since sequencer start is returned.

EmptyCount

This argument is a pointer to an int variable where the total number of empty buffer cycles since sequencer start is returned.

EXAMPLE

```
#include “tpmc551api.h”

TPMC551_HANDLE   hdl;
TPMC551_STATUS   result;
int              OperatingState;
int              status;
int              CycleCount;
int              UnderflowCount;
int              EmptyCount;

/* Read sequencer status and statistic information */

result = tpmc551SeqStatus(hdl, &OperatingState, &status, &CycleCount,
                          &UnderflowCount, &EmptyCount);
if (result != TPMC551_OK)
{
     /* handle error */
}
```

RETURNS

On success, TPMC551_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC551_ERR_INVALID_HANDLE	The specified TPMC551_HANDLE is invalid.

3 Appendix

3.1 Debugging and Diagnostic

The TPMC551 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 TPMC551 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_TPMC551_SHOW in tpmc551drv.c

The tpmc551Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TPMC551 modules and device statistics.

If TPMC551 modules were probed but no devices were created it may helpful to enable debugging code inside the driver code by defining the macro TPMC551_DEBUG in tpmc551drv.c.

In contrast to VxBus TPMC551 devices, legacy TPMC551 devices must be created "manually". This will be done with the first call to the tpmc551Open API function.

```
-> tpmc551Show
Probed Modules:
    [0] TPMC551: Bus=4, Dev=1, DevId=0x9050, VenId=0x10b5, Init=OK, vxDev=0x5380

Associated Devices:
    [0] TPMC551:     /tpmc551/0

Device Statistics:
    /tpmc551/0:
        Open Count            = 0
        Sequencer Cycle Count = 0
        Channels Output Range and Correction-Data (Offset/Gain):
            #1 [-10V... +10V] -   -11/45
            #2 [-10V... +10V] -    10/34
            #3 [-10V... +10V] -   -12/42
            #4 [-10V... +10V] -   -13/48
            #5 [  0V... +10V] -   -19/29
            #6 [  0V... +10V] -   -12/23
            #7 [  0V... +10V] -   -15/27
            #8 [  0V... +10V] -   -13/29
```