The Embedded I/O Company

TEWS TECHNOLOGIES

TXMC590-SW-42

VxWorks Device Driver

16 Channel Thermo-/Strain Measurement

Version 1.1.x

User Manual

Issue 1.1.0

May 2016

powerBridge Computer

Ehlbeek 15a
30938 Burgwedel
fon 05139-9980-0
fax 05139-9980-49

www.powerbridge.de
info@powerbridge.de

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

TXMC590-SW-42

VxWorks Device Driver

16 Channel Thermo-/Strain Measurement

Supported Modules:
TXMC590

Issue	Description	Date
1.0.0	First Issue	December 7, 2015
1.1.0	txmc590ConfigureChannel: TXMC590_NONE_CJ_SENSOR added txmc590CalibrateStrain() removed txmc590Calibrate() added	May 26, 2016

Table of Contents

1 Introduction

The TXMC590-SW-42 VxWorks device driver software allows the operation of the supported modules conforming to the VxWorks I/O system specification.

The TXMC590-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.

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

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

The TXMC590 device driver supports the following features:

- Configure channels
- Read temperature value (RTD/Thermocouple)
- Read strain value (Strain gauge)
- Calibrate strain gauge channel (Strain gauge)
- Configure cold junction channel
- Load configuration tables for custom devices

The TXMC590 device driver supports the modules listed below:

TXMC590	16 channel thermo-/strain measurement board	(XMC)

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

TXMC590 User Manual

2 Installation

Following files are located on the distribution media:

Directory path ‘TXMC590-SW-42’:

TXMC590-SW-42-1.1.0.pdf	PDF copy of this manual
TXMC590-SW-42-VXBUS.zip	Zip compressed archive with VxBus driver sources
TXMC590-SW-42-LEGACY.zip	Zip compressed archive with legacy driver sources
ChangeLog.txt	Release history
Release.txt	Release information

The archive TXMC590-SW-42-VXBUS.zip contains the following files and directories:

Directory path ‘./tews/txmc590’:

txmc590drv.c	TXMC590 device driver source
txmc590def.h	TXMC590 driver include file
txmc590.h	TXMC590 include file for driver and application
txmc590api.c	TXMC590 API file
txmc590api.h	TXMC590 API include file
Makefile	Driver Makefile
40txmc590.cdf	Component description file for VxWorks development tools
txmc590.dc	Configuration stub file for direct BSP builds
txmc590.dr	Configuration stub file for direct BSP builds
include/tvxbHal.h	Hardware dependent interface functions and definitions
apps/txmc590exa.c	Example application

The archive TXMC590-SW-42-LEGACY.zip contains the following files and directories:

Directory path ‘./txmc590’:

txmc590drv.c	TXMC590 device driver source
txmc590def.h	TXMC590 driver include file
txmc590.h	TXMC590 include file for driver and application
txmc590pci.c	TXMC590 device driver source for x86 based systems
txmc590api.c	TXMC590 API file
txmc590api.h	TXMC590 API include file
txmc590exa.c	Example application
include/tdhal.h	Hardware dependent interface functions and definitions

2.1 Legacy vs. VxBus Driver

In later VxWorks 6.x releases, the old VxWorks 5.x legacy device driver model was replaced by VxBus-enabled device drivers. Legacy device drivers are tightly coupled with the BSP and the board hardware. The VxBus infrastructure hides all BSP and hardware differences under a well-defined interface, which improves the portability and reduces the configuration effort. A further advantage is the improved performance of API calls by using the method interface and bypassing the VxWorks basic I/O interface.

VxBus-enabled device drivers are the preferred driver interface for new developments.

The checklist below will help you to make a decision which driver model is suitable and possible for your application:

Legacy Driver	VxBus Driver
▪ VxWorks 5.x releases ▪ VxWorks 6.5 and earlier releases ▪ VxWorks 6.x releases without VxBus PCI bus support	▪ VxWorks 6.6 and later releases with VxBus PCI bus ▪ SMP systems (only the VxBus driver is SMP safe!) ▪ 64-bit systems (only the VxBus driver is 64-bit compatible)

TEWS TECHNOLOGIES recommends not using the VxBus Driver before VxWorks release 6.6. In previous releases required header files are missing and the support for 3rd-party drivers may not be available.

2.2 VxBus Driver Installation

Because Wind River doesn't provide a standard installation method for 3rd party VxBus device drivers the installation procedure needs to be done manually.

In order to perform a manual installation extract all files from the archive TXMC590-SW-42-VXBUS.zip to the typical 3rd party directory *installDir/vxworks-6.x/target/3rdparty* (whereas *installDir* must be substituted by the VxWorks installation directory).

After successful installation the TXMC590 device driver is located in the vendor and driver-specific directory *installDir/vxworks-6.x/target/3rdparty/tews/txmc590.*

At this point the TXMC590 driver is not configurable and cannot be included with the kernel configuration tool in a Wind River Workbench project. To make the driver configurable the driver library for the desired processer (CPU) and build tool (TOOL) must be built in the following way:

(1) Open a VxWorks development shell (e.g. C:\WindRiver\wrenv.exe -p vxworks-6.7)

(2) Change into the driver installation directory
installDir/vxworks-6.x/target/3rdparty/tews/txmc590

(3) Invoke the build command for the required processor and build tool
make CPU=cpuName TOOL=tool

For Windows hosts this may look like this:

```
C:> cd \WindRiver\vxworks-6.7\target\3rdparty\tews\txmc590
C:> make CPU=PENTIUM4 TOOL=diab
```

To compile SMP-enabled libraries, the argument VXBUILD=SMP must be added to the command line

```
C:> make CPU=PENTIUM4 TOOL=diab VXBUILD=SMP
```

To build 64-bit libraries, the argument VXBUILD=LP64 must be added to the command line

```
> make CPU=CORE TOOL=gnu VXBUILD=LP64
```

For 64-bit SMP-enabled libraries a build command may look like this

```
> make CPU=CORE TOOL=gnu VXBUILD="LP64 SMP"
```

To integrate the TXMC590 driver with the VxWorks development tools (Workbench), the component configuration file *40txmc590.cdf* must be copied to the directory *installDir/vxworks-6.x/target/config/comps/VxWorks*.

```
C:> cd \WindRiver\vxworks-6.7\target\3rdparty\tews\txmc590
C:> copy 40txmc590.cdf \Windriver\vxworks-6.7\target\config\comps\vxWorks
```

In VxWorks 6.7 and newer releases the kernel configuration tool scans the CDF file automatically and updates the *CxrCat.txt* cache file to provide component parameter information for the kernel configuration tool as long as the timestamp of the copied CDF file is newer than the one of the *CxrCat.txt*. If your copy command preserves the timestamp, force to update the timestamp by a utility, such as *touch*.

In earlier VxWorks releases the CxrCat.txt file may not be updated automatically. In this case, remove or rename the original *CxrCat.txt* file and invoke the make command to force recreation of this file.

```
C:> cd \Windriver\vxworks-6.7\target\config\comps\vxWorks
C:> del CxrCat.txt
C:> make
```

After successful completion of all steps above and restart of the Wind River Workbench, the TXMC590 driver can be included in VxWorks projects by selecting the *"TEWS TXMC590 Driver"* and *"TEWS TXMC590 API"* components in the *"hardware (default) - Device Drivers"* folder with the kernel configuration tool.

2.2.1 Direct BSP Builds

In development scenarios with the direct BSP build method without using the Workbench or the vxprj command-line utility, the TXMC590 configuration stub files must be copied to the directory *installDir/vxworks-6.x/target/config/comps/src/hwif.* Afterwards the *vxbUsrCmdLine.c* file must be updated by invoking the appropriate make command.

```
C:> cd \WindRiver\vxworks-6.7\target\3rdparty\tews\txmc590
C:> copy txmc590.dc \Windriver\vxworks-6.7\target\config\comps\src\hwif
C:> copy txmc590.dr \Windriver\vxworks-6.7\target\config\comps\src\hwif

C:> cd \Windriver\vxworks-6.7\target\config\comps\src\hwif
C:> make vxbUsrCmdLine.c
```

2.3 Legacy Driver Installation

2.3.1 Include Device Driver in VxWorks Projects

For including the TXMC590-SW-42 device driver into a VxWorks project (e.g. Tornado IDE or Workbench) follow the steps below:

(1) Extract all files from the archive TXMC590-SW-42-LEGACY.zip to your project directory.

(2) Add the device drivers C-files to your project.
Make a right click to your project in the 'Workspace' window and use the 'Add Files ...' topic.
A file select box appears, and the driver files in the txmc590 directory can be selected.

(3) Now the driver is included in the project and will be built with the project.

For a more detailed description of the project facility please refer to your VxWorks User's Guide (e.g. Tornado, Workbench, etc.)

2.3.2 Special Installation for Intel x86 based Targets

The TXMC590 device driver is fully adapted for Intel x86 based targets. This is done by conditional compilation directives inside the source code and controlled by the VxWorks global defined macro **CPU_FAMILY**. If the content of this macro is equal to *I80X86* special Intel x86 conforming code and function calls will be included.

The second problem for Intel x86 based platforms can't be solved by conditional compilation directives. Due to the fact that some Intel x86 BSP's doesn't map PCI memory spaces of devices which are not used by the BSP, the required device memory spaces can't be accessed.

To solve this problem a MMU mapping entry has to be added for the required TXMC590 PCI memory spaces prior the MMU initialization (*usrMmuInit()*) is done.

The C source file **txmc590pci.c** contains the function *txmc590PciInit()*. This routine finds out all TXMC590 devices and adds MMU mapping entries for all used PCI memory spaces. Please insert a call to this function after the PCI initialization is done and prior to MMU initialization (*usrMmuInit()*).

The right place to call the function *txmc590PciInit()* is at the end of the function *sysHwInit()* in **sysLib.c** (it can be opened from the project *Files* window).

```
txmc590PciInit();
```

Be sure that the function is called prior to MMU initialization otherwise the TXMC590 PCI spaces remains unmapped and an access fault occurs during driver initialization.

Modifying the sysLib.c file will change the sysLib.c in the BSP path. Remember this for future projects and recompilations.

2.3.3 System Resource Requirement

The table gives an overview over the system resources that will be needed by the driver.

Resource	Driver requirement	Devices requirement
Memory	application dependent[1)] (generally < 5 KB)	< 1 KB
Stack	< 1 KB	---
Semaphores	2	1

1) Depends on Size of Table File (correct files have a maximum size of 4 KB)

Memory and Stack usage may differ from system to system, depending on the used compiler and its setup.

The following formula shows the way to calculate the common requirements of the driver and devices.

*<total requirement> = <driver requirement> + (<number of devices> * <device requirement>)*

The maximum usage of some resources is limited by adjustable parameters. If the application and driver exceed these limits, increase the according values in your project.

3 API Documentation

3.1 General Functions

3.1.1 txmc590Open

NAME

txmc590Open – opens a device

SYNOPSIS

```
TXMC590_HANDLE txmc590Open
(
      char          *DeviceName
)
```

DESCRIPTION

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

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TXMC590 device is named “/txmc590/0”, the second is named “/txmc590/1”, and so on.

EXAMPLE

```
#include “txmc590api.h”

TXMC590_HANDLE     hdl;

/*
** open the specified device
*/
hdl = txmc590Open(“/txmc590/0”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

The error codes are stored in *errno.*

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

3.1.2 txmc590Close

NAME

txmc590Close – closes a device

SYNOPSIS

```
TXMC590_STATUS txmc590Close
(
      TXMC590_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 "txmc590api.h"

TXMC590_HANDLE      hdl;
TXMC590_STATUS      result;

/*
** close the device
*/
result = txmc590Close(hdl);
if (result != TXMC590_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 txmc590GetModuleInfo

NAME

txm590GetModuleInfo – get information of the module

SYNOPSIS

```
TXMC590_STATUS txmc590GetModuleInfo
(
    TXMC590_HANDLE      hdl,
    TXMC590_INFO_BUF    *pModuleInfo
)
```

DESCRIPTION

This function returns information about the module, including PCI header as well as the PCI localization.

PARAMETERS

hdl

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

pModuleInfo

This argument is a pointer to the structure TXMC590_INFO_BUF that receives information of the module PCI header.

```
typedef struct
{
    unsigned int        pciBusNo;
    unsigned int        pciDevNo;
    unsigned int        pciFuncNo;
    unsigned short      vendorId;
    unsigned short      deviceId;
    unsigned short      subSystemId;
    unsigned short      subSystemVendorId;
    unsigned int        variant;
    unsigned int        fpgaRev;
    unsigned int        aducRev[16];
} TXMC590_INFO_BUF;
```

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

vendorId

PCI module vendor ID.

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module subsystem vendor ID

variant

Module variant (e.g. 10 for TXMC590-10)

fpgaRev

FPGA Code revision. Format: MMmmRRbb
MM - major version
mm - minor version
RR – revision
bb – reserved - internal build code

aducRev[]

ADuC-Firmware version for each channel. Format: MMmmRRbb
MM - major version
mm - minor version
RR – revision
bb – reserved - internal build code

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE          hdl;
TXMC590_STATUS          result;
TXMC590_INFO_BUF        infoBuf

...
```

```
...

/*
** get module information
*/
result = txmc590GetModuleInfo( hdl, &infoBuf );

if (result != TXMC590_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified device handle is invalid
TXMC590_ERR_INVAL	Specified pointer is invalid.

3.2 I/O Functions

3.2.1 txmc590ReadTemperature

NAME

txmc590ReadTemperature – Read temperature from specified channel

SYNOPSIS

```
TXMC590_STATUS txmc590ReadTemperature
(
        TXMC590_HANDLE      hdl,
        unsigned int        channel,
        unsigned int        unit,
        unsigned int        decPlaces,
        unsigned int        immediateRead,
        int                 *value
)
```

DESCRIPTION

This function reads a temperature from the specified channel. The unit of the returned value can be chosen.

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 which shall be read. Possible values are 0 up to 15.

unit

This argument specifies the unit of the returned temperature value. The following values are defined:

Value	Description
TXMC590_UNIT_RAW	The value read from the channel will be returned. The unit is defined in the configuration table of the channel.
TXMC590_UNIT_CELSIUS	The temperature will be returned in degrees Celsius.
TXMC590_UNIT_FAHRENHEIT	The temperature will be returned in degrees Fahrenheit.
TXMC590_UNIT_KELVIN	The temperature will be returned in Kelvin.

decPlaces

This argument specifies the decimal places of the returned value. This argument is not used if the argument unit is TXMC590_UNIT_RAW.
The table below shows how the conversion works.

Temperature	Specified Unit	Decimal Places	Returned Value
20.1638°C	°C	0	20
20.1638°C	°C	1	202
20.1638°C	°C	2	2016
20.1638°C	°C	3	20164
20.1638°C	°C	5	2016380
20.1638°C	K	2	29331
20.1638°C	°F	2	6829

immediateRead

This argument specifies if the function should wait for a new value. This flag is only used if the channel is configured in clock mode (TXMC590_TRMODE_CLOCK_xxx).

Value	Description
TXMC590_RDMODE_WAIT	The function will wait until the current clock period expires and the value is returned.
TXMC590_RDMODE_IMMEDIATE	The function returns the last converted value. The read is asynchronous to the update cycle.

value

This argument returns the temperature in the specified unit or if the raw unit is specified the value is returned in the unit defined in the configuration table.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE   hdl;
TXMC590_STATUS   result;
int              temperature;
double           fTemperature;
int              decPlaces = 3;        /* number of decimal places */

...
```

```
…

/* Wait for a new value */
result = txmc590ReadTemperature( hdl,
                                 1,
                                 TXMC590_UNIT_CELSIUS,
                                 decPlaces,
                                 TXMC590_RDMODE_WAIT,
                                 &temperature);
if (result != TXMC590_OK)
{
     /* handle error */
}
else
{
     fTemperature = temperature / pow(10, decPlaces);
     printf("Temperature (#1): %8.5f°C\n", fTemperature);
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid
TXMC590_ERR_INVAL	Invalid pointer or value specified
TXMC590_ERR_INVALID_CHANNEL	Invalid channel number selected
TXMC590_ERR_INVALID_MODE	Channel not configured to read temperature values
TXMC590_ERR_INVALID_TABLE	Invalid value detected in table
TXMC590_ERR_BUSY	Channel is busy
TXMC590_ERR_TIMEOUT	Access timed out
TXMC590_ERR_CONFIG	Channel announced a configuration error
TXMC590_ERR_CHANNEL	Channel announced a channel error

3.2.2 txmc590ReadStrain

NAME

txmc590ReadStrain – Read strain value from specified channel

SYNOPSIS

```
TXMC590_STATUS txmc590ReadStrain
(
        TXMC590_HANDLE      hdl,
        unsigned int        channel,
        unsigned int        decPlaces,
        unsigned int        immediateRead,
        int                 *value
)
```

DESCRIPTION

This function reads a strain value from the specified channel.

PARAMETERS

hdl

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

channel

This argument specifies the input channel which shall be read. Possible values are 0 up 15.

decPlaces

This argument specified the decimal places of the returned value.

immediateRead

This argument specifies if the function should wait for a new value. This flag is only used if the channel is configured in clock mode (TXMC590_TRMODE_CLOCK_xxx).

Value	Description
TXMC590_RDMODE_WAIT	The function will wait until the current clock period expires and the value is returned.
TXMC590_RDMODE_IMMEDIATE	The function returns the last converted value. The read is asynchronous to the update cycle.

value

This argument returns the strain value. The value unit is returned according to the specification of the configuration table.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE   hdl;
TXMC590_STATUS   result;
int              strain;

/* Do not wait for conversion, read last available value */
result = txmc590ReadStrain(hdl, 2, 3, TXMC590_RDMODE_IMMEDIATE, &strain);
if (result != TXMC590_OK)
{
     /* handle error */
}
else
{
     printf("Strain value (x1000) (#2): %d \n", strain);
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid
TXMC590_ERR_INVAL	Invalid pointer or value specified
TXMC590_ERR_INVALID_CHANNEL	Invalid channel number selected
TXMC590_ERR_INVALID_MODE	Channel not configured to read strain values
TXMC590_ERR_INVALID_TABLE	Invalid value detected in table
TXMC590_ERR_BUSY	Channel is busy
TXMC590_ERR_TIMEOUT	Access timed out
TXMC590_ERR_CONFIG	Channel announced a configuration error
TXMC590_ERR_CHANNEL	Channel announced a channel error

3.2.3 txmc590ReadColdJunction

NAME

txmc590ReadColdJunction – Read the cold junction temperature

SYNOPSIS

```
TXMC590_STATUS txmc590ReadColdJunction
(
        TXMC590_HANDLE      hdl,
        unsigned int        channel,
        unsigned int        unit,
        unsigned int        decPlaces,
        int                 *value
)
```

DESCRIPTION

This function reads a cold junction temperature (local input of TXMC590). The unit of the returned value can be chosen.

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 cold junction channel that shall be read. The following values are valid:

Value	Description
TXMC590_INTERNAL_CJ_SENSOR	Read the cold junction sensor locally mounted on the TXMC590.
TXMC590_EXTERNAL_CJ_SENSOR	Read the cold junction sensor externally connected to the TXMC590.

unit

This argument specifies the unit of the returned temperature value. The following values are defined::

Value	Description
TXMC590_UNIT_CELSIUS	The temperature will be returned in degrees Celsius.
TXMC590_UNIT_FAHRENHEIT	The temperature will be returned in degrees Fahrenheit.
TXMC590_UNIT_KELVIN	The temperature will be returned in Kelvin.
TXMC590_UNIT_RAW	Read raw value from TXMC590.

decPlaces

This argument specified the decimal places of the returned value. This argument is not used if the argument unit is TXMC590_UNIT_RAW.

value

This argument returns the cold junction temperature in the specified unit.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE   hdl;
TXMC590_STATUS   result;
int              temperature;
double           fTemperature;
int              decPlaces = 3;        /* number of decimal places */

result = txmc590ReadColdJunction(hdl,
                                 TXMC590_INTERNAL_CJ_SENSOR,
                                 TXMC590_UNIT_CELSIUS,
                                 decPlaces,
                                 &temperature);
if (result != TXMC590_OK)
{
     /* handle error */
}
else
{
     fTemperature = temperature / pow(10, decPlaces);
     printf("CJ-Temperature): %8.5f°C\n", fTemperature);
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid
TXMC590_ERR_INVAL	Invalid pointer or value specified
TXMC590_ERR_INVALID_CHANNEL	Invalid channel number selected
TXMC590_ERR_INVALID_MODE	Channel not configured

3.2.4 txmc590ConfigureChannel

NAME

txmc590ConfigureChannel – Configure an input channel

SYNOPSIS

```
TXMC590_STATUS txmc590ConfigureChannel
(
        TXMC590_HANDLE      hdl,
        unsigned int        channel,
        unsigned int        tableNo,
        unsigned int        conversionClockMode,
        unsigned short      conversionClockRate,
        unsigned int        coldJunctionSource
)
```

DESCRIPTION

This function configures the specified channel.

PARAMETERS

hdl

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

channel

This argument specifies the input channel which shall be read. Possible values are 0 up to 15.

tableNo

This argument specifies the table number that shall be used for the channel. The configuration table contains all necessary information which is necessary for the input channel to work. For more detailed information about the content of the table see TXMC590-UM.
There are predefined tables for the most common probes (0…15, see TXMC590 User Manual) and user defined tables (16…31), which can be loaded by the application for custom probes or special application.

conversionClockMode

This argument specifies the mode how conversions are initiated. The following modes are defined.

Value	Description
TXMC590_TRMODE_OFF	The channel is disabled
TXMC590_TRMODE_CONVERTONREAD	A new conversion will be started on a read access. The reading function will have to wait for completion of the conversion.
TXMC590_TRMODE_CLOCK_100US	The conversion will be started repeatedly with a defined rate. The rate is defined in conversionClockRate in steps of 100us. The reading function can use the value immediately.
TXMC590_TRMODE_CLOCK_100MS	The conversion will be started repeatedly with a defined rate. The rate is defined in conversionClockRate in steps of 1ms. The reading function can use the value immediately.
TXMC590_TRMODE_CLOCK_1S	The conversion will be started repeatedly with a defined rate. The rate is defined in conversionClockRate in steps of 1s. The reading function can use the value immediately.

conversionClockRate

This argument specifies the value of the conversion timer. The value is only used for conversionClockMode TXMC590_TRMODE_CLOCK_xxx.
The value is specified in units defined in conversionClockMode, e.g. a refresh cycle of 10ms will be generated with a value of 100 and conversionClockMode = TXMC590_TRMODE_CLOCK_100us.

coldJunctionSource

This argument specifies the channel that will be used to measure the cold junction temperature. This value will just be used for Thermocouple probes.

Value	Description
0..15	Use the appropriate channel as Cold Junction input. The channel must be configured before it is used as cold junction input.
TXMC590_INTERNAL_CJ_SENSOR	Use the cold junction sensor locally mounted on the TXMC590. The cold junction channel must be configured before it is used.
TXMC590_EXTERNAL_CJ_SENSOR	Use the cold junction sensor externally connected to the TXMC590. The cold junction channel must be configured before it is used.
TXMC590_NONE_CJ_SENSOR	Do not use a cold junction sensor. If this cold junction mode is selected, the temperature difference between hot and cold junction is measured.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE  hdl;
TXMC590_STATUS  result;

/* Use channel 1, whith table 5 update repeatedly every 10 seconds */
/* use the TXMC590 Cold Junction Temperature */
result = txmc590ConfigureChannel (    hdl,
                                  1,
                                  5,
                                  TXMC590_TRMODE_CLOCK_1S,
                                  10,
                                  TXMC590_INTERNAL_CJ_SENSOR);
if (result != TXMC590_OK)
{
     /* handle error */
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid.
TXMC590_ERR_INVALID_CHANNEL	Invalid channel number
TXMC590_ERR_INVALID_TABLE	Invalid table number specified, or content of table is invalid
TXMC590_ERR_CONFIG	Channel announced a configuration error
TXMC590_ERR_CHANNEL	Channel announced a channel error
TXMC590_ERR_INVAL	Invalid parameter pointer or value
TXMC590_ERR_TIMEOUT	Configuration timed out, configuration did not complete

3.2.5 txmc590ConfigureColdJunction

NAME

txmc590ConfigureColdJunction – Configure the TXMC590 Cold Junction channel

SYNOPSIS

```
TXMC590_STATUS txmc590ConfigureColdJunction
(
        TXMC590_HANDLE      hdl,
        unsigned int        channel,
        unsigned int        conversionClockMode,
        unsigned short      conversionClockRate
)
```

DESCRIPTION

This function configures the specified cold junction channel.

PARAMETERS

hdl

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

channel

This argument specifies the cold junction channel that shall be configured. The following values are valid:

Value	Description
TXMC590_INTERNAL_CJ_SENSOR	Configure the cold junction sensor locally mounted on the TXMC590.
TXMC590_EXTERNAL_CJ_SENSOR	Configure the cold junction sensor externally connected to the TXMC590.

conversionClockMode

This argument specifies the mode how conversions are initiated. The following modes are defined.

Value	Description
TXMC590_TRMODE_OFF	The channel is disabled
TXMC590_TRMODE_CLOCK_100US	The conversion will be started repeatedly with a defined rate. The rate is defined in conversionClockRate in steps of 100us. The reading function can use the value immediately.
TXMC590_TRMODE_CLOCK_100MS	The conversion will be started repeatedly with a defined rate. The rate is defined in conversionClockRate in steps of 1ms. The reading function can use the value immediately.
TXMC590_TRMODE_CLOCK_1S	The conversion will be started repeatedly with a defined rate. The rate is defined in conversionClockRate in steps of 1s. The reading function can use the value immediately.

conversionClockRate

This argument specifies the value of the conversion timer.
The value is specified in units defined in conversionClockMode, e.g. a refresh cycle of 10ms will be generated with a value of 100 and conversionClockMode = TXMC590_TRMODE_CLOCK_100us.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE  hdl;
TXMC590_STATUS  result;

/* Use internal cold junction sensor, update every 10 seconds */
result = txmc590ConfigureColdJunction (hdl,
                                       TXMC590_INTERNAL_CJ_SENSOR,
                                       TXMC590_TRMODE_CLOCK_1S,
                                       10);
if (result != TXMC590_OK)
{
     /* handle error */
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid.
TXMC590_ERR_INVALID_CHANNEL	Invalid channel number
TXMC590_ERR_INVAL	Invalid parameter pointer or value

3.2.6 txmc590Calibrate

NAME

txmc590Calibrate – Execute internal ADC calibration cycle

SYNOPSIS

```
TXMC590_STATUS txmc590Calibrate
(
      TXMC590_HANDLE     hdl,
      unsigned int       channel
)
```

DESCRIPTION

This function executes an ADC calibration cycle on the specified channel. The function blocks until the calibration is done. During the calibration, the channel cannot be used for measurements.

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 which shall be calibrated.
Possible values are 0 up to 15.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE  hdl;
TXMC590_STATUS  result;

result = txmc590Calibrate (hdl, 2);
if (result != TXMC590_OK)
{
     /* handle error */
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid.
TXMC590_ERR_INVALID_CHANNEL	Invalid channel number
TXMC590_ERR_CONFIG	Channel announced a configuration error
TXMC590_ERR_CHANNEL	Channel announced a channel error

3.2.7 txmc590LoadUserTable

NAME

txmc590LoadUserTable – Load a custom defined table

SYNOPSIS

```
TXMC590_STATUS txmc590LoadUserTable
(
        TXMC590_HANDLE      hdl,
        unsigned int        tableNo,
        char                *fileName
)
```

DESCRIPTION

This function loads a new table content into a specified custom configuration table space. This allows using custom and application adapted tables, e.g. for special probes or to use a more detailed resolution in the target temperature range.

For more detailed information about the content of the table see TXMC590 User Manual.

The table files must be available in simple hex format – a simple hex dump of the table space as specified in the TXMC590 User Manual, without address or checksum information.

PARAMETERS

hdl

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

tableNo

This argument specifies the table number that shall be loaded. Only user definable tables (16…31) are selectable.

fileName

This argument specifies file name (path) of the table file.

EXAMPLE

```
#include "txmc590api.h"

TXMC590_HANDLE  hdl;
TXMC590_STATUS  result;

result = txmc590LoadUserTable (hdl, 18, "./newTable.hex");
if (result != TXMC590_OK)
{
     /* handle error */
}
```

RETURNS

On success, TXMC590_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TXMC590_ERR_INVALID_HANDLE	The specified TXMC590_HANDLE is invalid.
TXMC590_ERR_INVAL	Invalid parameter pointer or value
TXMC590_ERR_INVFILE	Invalid file format
TXMC590_ERR_NOMEM	Cannot allocate memory
TXMC590_ERR_INVALID_TABLE	Invalid table number specified
TXMC590_ERR_NOFILE	Cannot access file
TXMC590_ERR_TIMEOUT	Loading table into, or from table space failed, or programming table into flash failed

4 Legacy I/O system functions

This chapter describes functions which are relevant only for the legacy TXMC590 device driver.

4.1 txmc590PciInit

NAME

txmc590PciInit – Generic PCI device initialization

SYNOPSIS

void txmc590PciInit()

DESCRIPTION

This function is required only for Intel x86 VxWorks platforms. The purpose is to setup the MMU mapping for all required TXMC590 PCI spaces (base address registers) and to enable the TXMC590 supported device for access.

The global variable *txmc590Status* 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 txmc590Status is equal to the number of mapped PCI spaces
0	No TXMC590 device found
< 0	Initialization failed. The value of (txmc590Status & 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 txmc590PciInit();

…

txmc590PciInit();
```

5 Debugging and Diagnostic

The TXMC590 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 TXMC590 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_TXMC590_SHOW in txmc590drv.c

The txmc590Show function (only if VxBus is used) displays detailed information about probed modules, assignment of devices respective device names to probed TXMC590 modules and device statistics.

If TXMC590 modules were probed but no devices were created it may helpful to enable debugging code inside the driver code by defining the macro TXMC590_DEBUG in txmc590drv.c.

In contrast to VxBus TXMC590 devices, legacy TXMC590 devices must be created “manually”. This will be done with the first call to the txmc590Open API function.

```
-> txmc590Show
Probed Modules:
    [0] TXMC590: Bus=3, Dev=0, DevId=0x924e, VenId=0x1498, Init=OK, vxDev=0x58d4b8

Associated Devices:
    [0] TXMC590: /txmc590/0

Device Statistics:
    /txmc590/0:
        open count = 0
        interrupt count = 0
value = 1 = 0x1
```