The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV005-SW-72

LynxOS Device Driver

6 Channel SSI, Incremental Encoder, Counter

Version 1.0.x

User Manual

Issue 1.0.0

March 2006

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

TEWS TECHNOLOGIES LLC
1 E. Liberty Street, Sixth Floor
Reno, Nevada 89504 / USA
Phone: +1 (775) 686 6077
Fax: +1 (775) 686 6024
e-mail: usasales@tews.com
www.tews.com

TDRV005-SW-72

6 Channel SSI, Incremental Encoder, Counter

LynxOS Device Driver

Supported Modules:
TPMC117

Issue	Description	Date
1.0.0	First Issue	March 30, 2006

Table of Contents

1 Introduction

The TDRV005-SW-72 LynxOS device driver allows the operation of the TPMC117 product family on LynxOS platforms with DRM based PCI interface.

The standard file (I/O) functions (open, close, ioctl) provide the basic interface for opening and closing a file descriptor and for performing device I/O and configuration operations.

The provided Application Programming Interface (API) should be used to access the TDRV005 specific functions. This API enhances the compatibility of a TDRV005 based application to different operating systems. The API itself makes usage of an abstraction layer, which adapts the different operating system entry calls. This results in a compatibility of the API too.

The TDRV005 device driver and its API support the following features:

- operate channels in SSI mode
  - setup and configure channel (SSI or SSI listen-only)
  - read SSI data
- operate channels in Counter mode
  - setup and configure channel
  - read counter data
  - setup preload value
  - load preload value into counter
  - reset counter
  - wait for MATCH event
  - wait for ControlMode event
- operate the onboard interval timer
  - setup and configure interval timer
  - start and stop interval timer
  - read interval timer value
  - wait for interval timer event
  - setup and use interval timer as trigger event for simultaneous multiple channel read
- enable and disable multiple channels simultaneously
- simultaneously read multiple channel values
- setup and load counter preload values simultaneously

The TDRV005-SW-72 supports the modules listed below:

TPMC117	6 Channel Interface	6 Channel SSI, Incremental Encoder, Counter

To get more information about the features and use of TDRV005 devices it is recommended to read the manuals listed below.

TPMC117 User manual

TPMC117 Engineering Manual

2 Installation

Following files are located in the directory TDRV005-SW-72 on the distribution media:

TDRV005-SW-72-1.0.0.pdf	This manual in PDF format
TDRV005-SW-72-SRC.tar	Device Driver and Example sources
Release.txt	Information about the Device Driver Release

The TAR archive TDRV005-SW-72-SRC.tar contains the following files and directories:

tdrv005.c	Driver source code
tdrv005.h	Definitions and data structures for driver and application
tdrv005def.h	Definitions and data structures for the driver
tdrv005_info.c	Device information definition
tdrv005_info.h	Device information definition header
tdrv005.cfg	Driver configuration file include
tdrv005.import	Linker import file
TCDI/tcdi_lib.h	Device Driver Interface header file (used by device driver)
TCDI/tcdi_lib.c	Device Driver Interface source file
TCDI/node.c	Linked-List management source file
TCDI/node.h	Linked-List management header file
TCOSI/tcosi_lib.h	Common Operating System Interface abstraction layer (used by API)
TCOSI/tcosi_lib.c	Common Operating System Interface abstraction layer (used by API)
tdrv005api.h	API include file
tdrv005api.c	API source file
Makefile	Device driver make file
example/tdrv005exa.c	Example application source
example/Makefile	Example application make file

In order to perform a driver installation first extract the TAR file to a temporary directory, then follow the steps below:

1. Create a new directory in the system drivers directory path /sys/drivers.xxx, where xxx represents the BSP that supports the target hardware.

   For example: /sys/drivers.pp_drm/tdrv005 or /sys/drivers.cpci_x86/tdrv005

2. Copy the following files to this directory:
   - tdrv005.c
   - tdrv005api.c
   - tdrv005def.h
   - tdrv005.import
   - Makefile

3. Create the sub-directories TCDI and TCOSI below the driver directory, and copy the following files into these directories:
   TCDI: node.c, node.h, tcdi_lib.c, tcdi_lib.h
   TCOSI: tcosi_lib.c

4. Copy tdrv005.h, tdrv005api.h and tcosi_lib.h to /usr/include/

5. Copy tdrv005_info.c to /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).

6. Copy tdrv005_info.h to /sys/dheaders/

7. Copy tdrv005.cfg to */sys/cfg.xxx/*, where xxx represents the BSP for the target platform. For example: /sys/cfg.ppc or /sys/cfg.x86 ....

2.1 Device Driver Installation

The two methods of driver installation are as follows:

- Static Installation
- Dynamic Installation (only native LynxOS systems)

2.1.1 Static Installation

With this method, the driver object code is linked with the kernel routines and is installed during system start-up.

2.1.1.1 Build the driver object

1. Change to the directory /sys/drivers.xxx/tdrv005, where xxx represents the BSP that supports the target hardware.
2. To update the library /sys/lib/libdrivers.a enter:

   ```
   make install
   ```

2.1.1.2 Create Device Information Declaration

1. Change to the directory /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).
2. Add the following dependencies to the Makefile

   ```
   DEVICE_FILES_all = ... tdrv005_info.x
   ```

   And at the end of the Makefile

   ```
   tdrv005_info.o:$(DHEADERS)/tdrv005_info.h
   ```

3. To update the library /sys/lib/libdevices.a enter:

   ```
   make install
   ```

2.1.1.3 Modify the Device and Driver Configuration File

In order to insert the driver object code into the kernel image, an appropriate entry in file CONFIG.TBL must be created.

1. Change to the directory /sys/lynx.os/ respective /sys/bsp.xxx, where xxx represents the BSP that supports the target hardware.
2. Create an entry at the end of the file CONFIG.TBL

   Insert the following entry at the end of this file.

   ```
   I:tdrv005.cfg
   ```

2.1.1.4 Rebuild the Kernel

1. Change to the directory /sys/lynx.os/ (/sys/bsp.xxx)
2. Enter the following command to rebuild the kernel:

   ```
   make install
   ```

3. Reboot the newly created operating system by the following command (not necessary for KDIs):

   ```
   reboot –aN
   ```

   The N flag instructs init to run mknod and create all the nodes mentioned in the new nodetab.

4. After reboot you should find the following new devices (depends on the device configuration): /dev/tdrv005a, /dev/tdrv005b, …

2.1.2 Dynamic Installation

This method allows you to install the driver after the operating system is booted. The driver object code is attached to the end of the kernel image and the operating system dynamically adds this driver to its internal structures. The driver can also be removed dynamically.

2.1.2.1 Build the driver object

1. Change to the directory /sys/drivers.xxx/tdrv005, where xxx represents the BSP that supports the target hardware.
2. To make the dynamic link-able driver enter :

   ```
   make dldd
   ```

2.1.2.2 Create Device Information Declaration

1. Change to the directory /sys/drivers.xxx/tdrv005, where xxx represents the BSP that supports the target hardware.
2. To create a device definition file for the major device (this works only on native systems)

   ```
   make t005info
   ```

3. To install the driver enter:

   ```
   drinstall –c tdrv005.obj
   ```

   If successful, drinstall returns a unique <driver-ID>
4. To install the major device enter:

   ```
   devinstall –c –d <driver-ID> t005info
   ```

   The <driver-ID> is returned by the drinstall command
5. To create nodes for the minor device enter:

   ```
   mknod /dev/tdrv005a c <major_no> 0
   ```

   The <major_no> is returned by the devinstall command.

If all steps are successfully completed, the TDRV005 is ready to use.

2.1.2.3 Uninstall dynamic loaded driver

To uninstall the TDRV005 device enter the following commands:

```
devinstall –u –c <device-ID>
drinstall –u <driver-ID>
```

2.1.3 Device Information Definition File

The device information definition contains information necessary to install the TDRV005 major device.

The implementation of the device information definition is done through a C structure, which is defined in the header file *tdrv005_info.h*.

This structure contains the following parameter:

PCIBusNumber Contains the PCI bus number at which the supported device is connected. Valid bus numbers are in range from 0 to 255.

PCIDeviceNumber Contains the device number (slot) at which the supported device is connected. Valid device numbers are in range from 0 to 31.

If both PCIBusNumber and PCIDeviceNumber are –1 then the driver will auto scan for supported devices. The first device found in the scan order will be allocated by the driver for this major device.

Already allocated devices can't be allocated twice. This is important to know if there are more than one TDRV005 major devices.

A device information definition is unique for every TDRV005 major device. The file *tdrv005_info.c* on the distribution media contains two device information declarations, **tdrv005a_info** for the first major device and **tdrv005b_info** for the second major device.

If the driver should support more than two major devices it is necessary to copy and paste an existing declaration and rename it with a unique name, for example **tdrv005c_info**, **tdrv005d_info** and so on.

It is also necessary to modify the device and driver configuration file, respectively the configuration include file *tdrv005.cfg*.

The following device declaration information uses the auto find method to detect a supported device on the PCI bus.

```
TDRV005_INFO tdrv005a_info = {

     -1,           /*  Auto find the device on any PCI bus   */
     -1,

};
```

2.1.4 Configuration File: CONFIG.TBL

The device and driver configuration file CONFIG.TBL contains entries for device drivers and its major and minor device declarations. Each time the system is rebuild, the config utility reads this file and produces a new set of driver and device configuration tables and a corresponding nodetab.

To install the TDRV005 driver and device into the LynxOS system, the configuration include file tdrv005.cfg must be included in the CONFIG.TBL (see also chapter 2.1.1.3).

The file tdrv005.cfg on the distribution disk contains the driver entry (*C:tdrv005:*\....) and two major device entries (*D:TDRV005 1:tdrv005a_info:: and D:TDRV005 2:tdrv005b_info::*).

If the driver should support more than one major device, the following entries for major devices must be enabled by removing the comment character (#). By copy and paste existing major and minor entries and renaming the new entries, it is possible to add any number of additional TDRV005 devices.

This example shows a driver entry with one major device and one minor device:

```
#    Format:
#    C:driver-name:open:close:read:write:select:control:install:uninstall
#    D:device-name:info-block-name:raw-partner-name
#    N:node-name:minor-dev

C:tdrv005:\
      :tdrv005open:tdrv005close:::\
      ::tdrv005ioctl:tdrv005install:tdrv005uninstall
D:TDRV005 1:tdrv005a_info::
N:tdrv005a:0
D:TDRV005 2:tdrv005b_info::
N:tdrv005b:0
```

The configuration above creates the following nodes in the /dev directory.

```
/dev/tdrv005a  /dev/tdrv005b
```

3 TDRV005 API Documentation

This TDRV005 Device Driver Application Programming Interface (API) enhances the compatibility of a TDRV005 based application to different operating systems. The API itself uses an abstraction layer called Common Operating System Interface (TCOSI), which hides the different operating system entry functions like open(), close(), read(), write() and ioctl() under a well-defined interface. This results in an operating system independent design of the API.

The TDRV005 API concept helps to provide applications on different operating systems and platforms with only a few changes to the application itself.

3.1 General Functions

3.1.1 tdrv005open()

NAME

tdrv005open() – opens a device.

SYNOPSIS

```
int tdrv005open
(
      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.

RETURN VALUE

If the function succeeds, the return value is an open handle called file descriptor to the specified device. If the function fails, a negative error code is returned.

ERRORS

TERR_INVALID_HANDLE_VALUE	The specified device does not exist.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;

/*
** open file descriptor to device
*/
FileDescriptor = tdrv005open( “/dev/tdrv005a” );
if (FileDescriptor < 0)
{
      /* handle open error */
}
```

3.1.2 tdrv005close()

NAME

tdrv005close() – closes a device.

SYNOPSIS

```
int tdrv005close
(
      int  FileDescriptor
);
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

FileDescriptor

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

RETURN VALUE

TEWS_OK if the device was closed successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_HANDLE_VALUE	Invalid file descriptor specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** close file descriptor to device
*/
result = tdrv005close( FileDescriptor );
if (result < 0)
{
      /* handle close error */
}
```

3.2 SSI Functions

3.2.1 tdrv005ssiSetup()

NAME

tdrv005ssiSetup() – sets up a channel for SSI operation.

SYNOPSIS

```
int tdrv005ssiSetup
(
      int             FileDescriptor,
      unsigned char   Channel,
      TD005_SSI_SETUP *Options
);
```

DESCRIPTION

This function sets up a specific channel to the provided SSI configuration. The function returns immediately to the caller after setting up the corresponding channel.

The SSI channel is not enabled by this command. This must be done by a subsequent call to tdrv005globalChannelEnable (see chapter 3.6.1).

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel which should be affected. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

Options

This value specifies the necessary configuration options in the structure *TD005_SSI_SETUP* with the following layout:

```
typedef struct
{
        TD005_SSI_MODE      Mode;
        unsigned char       NumberOfDataBits;
        TD005_SSI_CODING    Coding;
        unsigned char       ZeroBits;
```

```
        TD005_SSI_PARITY      Parity;
        unsigned char         ClockRate;
} TD005_SSI_SETUP;
```

MEMBERS

Mode

This value specifies the desired operation mode of the SSI interface. Possible values are:

Value	Description
TD005_MODE_STANDARD	Standard SSI operation
TD005_MODE_LISTENONLY	SSI interface operates in listen-only mode.

NumberOfDataBits

This value specifies the number of data bits to use. Possible values are between 1 and 32.

Coding

This value specifies the desired coding format. Possible values are:

Value	Description
TD005_CODING_BINARY	Binary coding is used.
TD005_CODING_GRAY	Gray coding is used, data is converted into binary.

ZeroBits

This value specifies the number of zero bits to use in combination with parity. Possible values are either 0 or 1.

Parity

This value specifies what kind of parity bit should be used. Possible values are:

Value	Description
TD005_PARITY_NONE	No parity bit is used.
TD005_PARITY_EVEN	An even parity bit is used.
TD005_PARITY_ODD	An odd parity bit is used.

ClockRate

This value specifies the clock rate for the encoder's serial clock speed. The clock can be programmed in steps of 1µs in the range of 1 to 15.

RETURN VALUE

TEWS_OK if the channel was configured successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
TD005_SSI_SETUP Options;

/*
** setup the counter with appropriate options
*/
Options.Mode                = TD005_MODE_STANDARD;
Options.NumberOfDataBits    = 32;
Options.Coding              = TD005_CODING_BINARY;
Options.ZeroBits            = 1;
Options.Parity              = TD005_PARITY_NONE;
Options.ClockRate           = 10;

result = tdrv005ssiSetup( FileDescriptor, TD005_CH0, &Options );
if (result < 0)
{
      /* handle configuration error */
}
```

3.2.2 tdrv005ssiRead()

NAME

tdrv005ssiRead() – reads the value of an SSI channel.

SYNOPSIS

```
int tdrv005ssiRead
(
      int             FileDescriptor,
      unsigned char   Channel,
      int             Timeout,
      unsigned long   *Data,
      unsigned long   *Status
);
```

DESCRIPTION

This function reads the value of the corresponding channel's data register. Only the number of previously configured data bits is valid. The function returns to the caller after the desired channel is read or the specified timeout occurred.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel which should be affected. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the data to be valid, *TD005_WAIT_FOREVER* must be specified.

Data

This parameter points to an unsigned long value where the data register content is stored.

Status

This parameter points to an unsigned long value where the status register content is stored.

RETURN VALUE

TEWS_OK if the read operation was successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_BUSY	Channel is not configured to SSI mode, or there is another job in progress.
TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
unsigned long ssiValue;
unsigned long ssiStatus;

/*
** read the current counter value
*/
result = tdrv005ssiRead( FileDescriptor,
                         TD005_CH0,
                         TD005_WAIT_FOREVER,
                         &ssiValue,
                         &ssiStatus );
if (result == TEWS_OK)
{
      printf( SSI Value  = 0x%08lX\n”, ssiValue );
      printf( SSI Status = 0x%08lX\n”, ssiStatus );

}
```

3.3 Counter Functions

3.3.1 tdrv005counterSetup()

NAME

tdrv005counterSetup() – sets up a channel for counter operation.

SYNOPSIS

```
int tdrv005counterSetup
(
      int                   FileDescriptor,
      unsigned char         Channel,
      TD005_COUNTER_SETUP   *Options
);
```

DESCRIPTION

This function sets up a specific channel to the provided counter configuration. The function returns immediately to the caller after setting up the corresponding channel.

The counter channel is not enabled by this command. This must be done by a subsequent call to tdrv005globalChannelEnable (see chapter 3.6.1).

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel which should be affected. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

Options

This value specifies the necessary configuration options in the structure *TD005_COUNTER_SETUP* with the following layout:

```
typedef struct
{
        unsigned char         Polarity;
        TD005_CNT_INPUT       InputMode;
        TD005_CNT_INDEX       IndexControlMode;
        TD005_CNT_SCM         SpecialCountMode;
        TD005_CNT_CLKDIV      ClockPrescaler;
} TD005_COUNTER_SETUP;
```

MEMBERS

Polarity

This value specifies the input polarity of the specified channel. The Input Polarity Control can be used to adapt the input to the input source polarity of A, B and I. Use the following predefined values to generate an OR'ed polarity value.

Value	Description
TD005_POLARITY_A_LOW	low-active signal, default is high-active
TD005_POLARITY_B_LOW	low-active signal, default is high-active
TD005_POLARITY_I_LOW	low-active signal, default is high-active

InputMode

The Input Mode determines the input source and how the counter interprets these input signals. Possible values are:

Value	Description	Input Source
TD005_INPUT_TIMER_UP	Timer mode Up	internal clock prescaler
TD005_INPUT_TIMER_DOWN	Timer mode Down	internal clock prescaler
TD005_INPUT_DIRECTION	Direction count	Input A & Input B
TD005_INPUT_UPDOWN	Up/Down count	Input A & Input B
TD005_INPUT_QUADRATURE_1X	Quadrature count 1x	Input A & Input B
TD005_INPUT_QUADRATURE_2X	Quadrature count 2x	Input A & Input B
TD005_INPUT_QUADRATURE_4X	Quadrature count 4x	Input A & Input B

IndexControlMode

The Index Control Mode determines how the counter interprets events on the I-input. Possible values are:

Value	Description
TD005_ICM_NO_INDEX_CONTROL	no I-control
TD005_ICM_LOAD_ON_INDEX	load on index signal
TD005_ICM_LATCH_ON_INDEX	latch on index signal
TD005_ICM_GATE_ON_INDEX	gate on index signal
TD005_ICM_RESET_ON_INDEX	reset on index signal
TD005_ICM_REFERENCE_MODE	reference mode (quadrature input mode only)
TD005_ICM_AUTO_REFERENCE_MODE	auto-reference mode (quadrature input mode only)
TD005_ICM_INDEX_MODE	index mode (quadrature input mode only)

SpecialCountMode

This value specifies the desired special count mode. Possible values are:

Value	Description
TD005_SCM_CYCLING_COUNTER	No special count mode, cycling counter.
TD005_SCM_DIVIDE_BY_N	Divide-by-N mode.
TD005_SCM_SINGLE_CYCLE	Single cycle mode.

ClockPrescaler

This value specifies the internal clock prescaler to be used. Possible values are:

Value	Description
TD005_CLKDIV_1X	Prescaler 1x, 32 MHz clock
TD005_CLKDIV_2X	Prescaler 2x, 16 MHz clock
TD005_CLKDIV_4X	Prescaler 4x, 8 MHz clock
TD005_CLKDIV_8X	Prescaler 8x, 4 MHz clock

RETURN VALUE

TEWS_OK if the counter was configured successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
TD005_COUNTER_SETUP    Options;

/*
** setup the counter with appropriate options
*/
Options.Polarity            = TD005_POLARITY_A_LOW |
                              TD005_POLARITY_B_LOW |
                              TD005_POLARITY_I_LOW;
Options.InputMode           = TD005_INPUT_UPDOWN;
Options.IndexControlMode    = TD005_ICM_NO_INDEX_CONTROL;
Options.SpecialCountMode    = TD005_SCM_CYCLING_COUNTER;
Options.ClockPrescaler      = TD005_CLKDIV_8X;

result = tdrv005counterSetup( FileDescriptor, TD005_CH0, &Options );
if (result < 0)
{
      /* handle configuration error */
}
```

3.3.2 tdrv005counterRead()

NAME

tdrv005counterRead() – reads the value of a counter channel.

SYNOPSIS

```
int tdrv005counterRead
(
      int             FileDescriptor,
      unsigned char   Channel,
      unsigned long   *Data,
      unsigned long   *Status
);
```

DESCRIPTION

This function reads the value of the corresponding channel's data register. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel on which the specified event should occur. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

Data

This parameter points to an unsigned long value where the data register content is stored.

Status

This parameter points to an unsigned long value where the status register content is stored.

RETURN VALUE

TEWS_OK if the read operation was successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Channel is not configured to counter mode.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
unsigned long CounterValue, Status;

/*
** read the current counter value
*/
result = tdrv005counterRead( FileDescriptor,
                             TD005_CH0,
                             &CounterValue,
                             &Status );
if (result == TEWS_OK)
{
      printf( Counter Value = 0x%08lX\n”, CounterValue );
      printf( Status  Value = 0x%08lX\n”, Status );

}
```

3.3.3 tdrv005counterPreloadSet()

NAME

tdrv005counterPreloadSet() – sets the preload register of a counter channel.

SYNOPSIS

```
int tdrv005counterPreloadSet
(
      int             FileDescriptor,
      unsigned char   Channel,
      unsigned long   PreloadValue
);
```

DESCRIPTION

Set the counter's preload register to the supplied value. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel which should be affected. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

PreloadValue

This value specifies the new value of the channel's preload register.

RETURN VALUE

TEWS_OK if the counter preload register was set successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Channel is not configured to counter mode.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
unsigned long PreloadValue;

/*
** load counter value
*/
PreloadValue = 0x12345678;
result = tdrv005counterPreloadSet( TD005_CH0, PreloadValue );
if (result < 0)
{
      /* handle error */
}
```

3.3.4 tdrv005counterLoad()

NAME

tdrv005counterLoad() – loads the preload register into the counter channel.

SYNOPSIS

```
int tdrv005counterLoad
(
      int             FileDescriptor,
      unsigned char   Channel
);
```

DESCRIPTION

Load the counter to the previously specified value of the counter preload register. The function returns immediately to the caller. To simultaneously load multiple channels, please refer to chapter 3.6.4.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel which should be affected. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

RETURN VALUE

TEWS_OK if the counter was loaded successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Channel is not configured to counter mode.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** load counter value
*/
result = tdrv005counterLoad( FileDescriptor, TD005_CH0 );
if (result < 0)
{
      /* handle error */
}
```

3.3.5 tdrv005counterReset()

NAME

tdrv005counterReset() – resets the counter channel.

SYNOPSIS

```
int tdrv005counterReset
(
      int             FileDescriptor,
      unsigned char   Channel
);
```

DESCRIPTION

Reset the counter value of the specified channel. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel which should be affected. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

RETURN VALUE

TEWS_OK if the counter was reset successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Channel is not configured to counter mode.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** reset counter value
*/
result = tdrv005counterReset( FileDescriptor, TD005_CH0 );
if (result < 0)
{
      /* handle error */
}
```

3.3.6 tdrv005counterWaitMatch()

NAME

tdrv005counterWaitMatch() – waits for a counter-match event.

SYNOPSIS

```
int tdrv005counterWaitMatch
(
      int             FileDescriptor,
      unsigned char   Channel,
      unsigned long   CompareValue,
      int             Timeout
);
```

DESCRIPTION

Waits until the counter value matches the provided counter compare value. The function returns to the caller if the counter matches the provided compare value, or the specified timeout occurred.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel on which the specified event should occur. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

CompareValue

This parameter specifies the value to which the counter should be compared.

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TD005_WAIT_FOREVER must be specified.

RETURN VALUE

TEWS_OK if the counter event occurred successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_TIMEOUT	The event has not occurred, timeout.
TERR_BUSY	Channel is not configured to counter mode, or a counter-match job is already pending.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
unsigned long MatchValue;

/*
** wait indefinitely for counter match event
*/
MatchValue = 0x12345678;
result = tdrv005counterWaitMatch( FileDescriptor,
                                  TD005_CH0,
                                  MatchValue,
                                  TD005_WAIT_FOREVER );
if (result < 0)
{
      /* handle error */
}
```

3.3.7 tdrv005counterWaitControlModeEvent()

NAME

tdrv005counterWaitControlModeEvent() – waits for a counter-control-mode event.

SYNOPSIS

```
int tdrv005counterWaitControlModeEvent
(
      int             FileDescriptor,
      unsigned char   Channel,
      int             Timeout
);
```

DESCRIPTION

Wait for the control mode event of the counter. The function returns to the caller if the configured control mode event or the specified timeout occurred.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel on which the specified event should occur. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TD005_WAIT_FOREVER must be specified.

RETURN VALUE

TEWS_OK if the counter event occurred successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_TIMEOUT	The event has not occurred, timeout.
TERR_BUSY	Channel is not configured to counter mode, or a counter-control job is already pending.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** wait indefinitely for counter control mode event
*/
result = tdrv005counterWaitControlModeEvent( FileDescriptor,
                                             TD005_CH0,
                                             TD005_WAIT_FOREVER );
if (result < 0)
{
      /* handle error */
}
```

3.4 Timer Functions

3.4.1 tdrv005timerSetup()

NAME

tdrv005timerSetup() – sets up the timer.

SYNOPSIS

```
int tdrv005timerSetup
(
      int                   FileDescriptor,
      TD005_CNT_CLKFRQ      ClockFrequency,
      unsigned short        PreloadValue
);
```

DESCRIPTION

This function sets up the onboard timer to the provided configuration. The function returns immediately to the caller. The interval timer remains stopped after a call to this function.

PARAMETERS

FileDescriptor

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

ClockFrequency

This value specifies the clock frequency used as clock source for the timer. Possible values are:

Value	Description
TD005_CLKFRQ_1MHZ	1 MHz clock
TD005 CLKFRQ 2MHZ	2 MHz clock
TD005_CLKFRQ_4MHZ	4 MHz clock
TD005_CLKFRQ_8MHZ	8 MHz clock

PreloadValue

This value specifies the preload value of the timer. If the interval timer is running, this value is loaded automatically every time the timer expires. The preload value is of 16bit width.

RETURN VALUE

TEWS_OK if the timer was configured successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** setup the interval timer with an interrupt frequency of 100 Hz
*/
result = tdrv005timerSetup( FileDescriptor,
                            TD005_CLKFRQ_1MHZ,
                            10000 );
if (result < 0)
{
      /* handle error */
}
```

3.4.2 tdrv005timerStart()

NAME

tdrv005timerStart() – starts the timer.

SYNOPSIS

```
int tdrv005timerStart
(
      int FileDescriptor
);
```

DESCRIPTION

Start the previously configured timer. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

RETURN VALUE

TEWS_OK if the timer was started successfully, otherwise a negative error code.

ERRORS

TERR_NO_CONFIGURATION	The timer was not configured properly.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** start the previously configured interval timer
*/
result = tdrv005timerStart( FileDescriptor );
if (result < 0)
{
      /* handle error */
}
```

3.4.3 tdrv005timerStop()

NAME

tdrv005timerStop() – stops the timer.

SYNOPSIS

```
int tdrv005timerStop
(
      int FileDescriptor
);
```

DESCRIPTION

Stop the previously configured timer. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

RETURN VALUE

TEWS_OK if the timer was stopped successfully, otherwise a negative error code.

ERRORS

TERR_NO_CONFIGURATION	The timer was not configured properly.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** stop the interval timer
*/
result = tdrv005timerStop( FileDescriptor );
if (result < 0)
{
      /* handle error */
}
```

3.4.4 tdrv005timerRead()

NAME

tdrv005timerRead() – reads the current timer value.

SYNOPSIS

```
int tdrv005timerRead(
      int             FileDescriptor,
      unsigned short  *Data
);
```

DESCRIPTION

Read the current value of the timer. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Data

This parameter points to an unsigned short value where the data register content is stored.

RETURN VALUE

TEWS_OK if the read operation was successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
unsigned short TimerValue;

/*
** read the current interval timer value
*/
result = tdrv005timerRead( FileDescriptor, &TimerValue );
if (result == TEWS_OK)
{
      printf( Timer Value = 0x%.4X\n”, TimerValue );
}
```

3.4.5 tdrv005timerWait()

NAME

tdrv005timerWait() – waits for a timer event.

SYNOPSIS

```
int tdrv005timerWait
(
      int FileDescriptor,
      int Timeout
);
```

DESCRIPTION

Wait for the timer event or the specified timeout to occur. The function returns to the caller if the timer has expired, or the specified timeout occurred.

PARAMETERS

FileDescriptor

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

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TD005_WAIT_FOREVER must be specified.

RETURN VALUE

TEWS_OK if the timer event occurred successfully, otherwise a negative error code.

ERRORS

TERR_NOT_RUNNING	The timer is not running.
TERR_NO_CONFIGURATION	The timer was not configured properly.
TERR_TIMEOUT	The timer event has not occurred, timeout.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** wait indefinitely for an interval timer event
*/
result = tdrv005timerWait( FileDescriptor, TD005_WAIT_FOREVER );
if (result < 0)
{
      /* handle error */
}
```

3.4.6 tdrv005timerMultipleChannelReadSetup()

NAME

tdrv005timerMultipleChannelReadSetupt() – sets up channels for multiple read.

SYNOPSIS

```
int tdrv005timerMultipleChannelReadSetup
(
      int             FileDescriptor,
      unsigned char   Channel
);
```

DESCRIPTION

Configure specified channels for simultaneous sampling triggered by the timer. The function returns immediately to the caller.

Channels configured to SSI listen-only mode may not be used with this function.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channels which should be read simultaneously. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Multiple channels may be OR'ed to one value.

RETURN VALUE

TEWS_OK if the multiple-channel-read operation was configured successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified, or a specified channel is not configured properly.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Specified channel is busy with an SSI job.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** setup channels 0+5 for simultaneous sampling triggered by timer
*/
result = tdrv005timerMultipleChannelReadSetup( FileDescriptor,
                                                TD005_CH0 | TD005_CH5 );
if (result < 0)
{
      /* handle error */
}
```

3.4.7 tdrv005timerMultipleChannelReadWait()

NAME

tdrv005timerMultipleChannelReadWait() – waits for the simultaneous data timer event.

SYNOPSIS

```
int tdrv005timerMultipleChannelReadWait
(
      int                   FileDescriptor,
      TD005_MULTIPLEVALUES  *MultipleValues,
      unsigned long         *Timestamp,
      int                   *MoreDataAvailable,
      int                   Timeout
);
```

DESCRIPTION

Return the values of simultaneously sampled channels. An array to store all channel values must be supplied to this function. The function waits for the timer event to occur on which the previously configured channels are sampled, or the specified timeout occurred. Before using this function the timer must be configured properly.

PARAMETERS

FileDescriptor

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

MultipleValues

This is a pointer to a TD005_MULTIPLEVALUES structure, where the read values are stored. Note that only the previously configured channels return valid data, other data entries must be ignored. The TD005_MULTIPLEVALUES structure has the following layout:

```
typedef struct
{
       TD005_VALUE_BUF Channel[6];
} TD005_MULTIPLEVALUES;
```

MEMBERS

Channel

This parameter is an array of a TD005_VALUE_BUF structure which holds the returned channel data. The value for channel 0 is returned at array index 0, channel 5's value is located at array index 5. The TD005_VALUE_BUF structure has the following layout:

```
typedef struct
{
         unsigned long   Value;
         unsigned long   Status;
} TD005_VALUE_BUF;
```

MEMBERS

Value

This parameter holds the returned channel value.

Status

This parameter holds the returned channel status.

Timestamp

This is a pointer to an unsigned long value where the number of occurred timer interrupts is returned.

MoreDataAvailable

This is a pointer to a boolean integer value. The value is TRUE if additional data is available. This might happen if the timer event triggering the multiple-channel-read operation appears too fast. An additional call to td005timerMultipleChannelReadWait must be performed.

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TD005_WAIT_FOREVER must be specified.

RETURN VALUE

TEWS_OK if the timer event triggering the multiple-channel-read operation occurred successfully and data is available, otherwise a negative error code.

ERRORS

TERR_NOT_RUNNING	The timer is not running.
TERR_NO_CONFIGURATION	The timer was not configured properly.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_TIMEOUT	The timer event has not occurred, timeout.
TERR_BUSY	A MultipleChannelRead job is already pending.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
int MoreDataAvailable;
unsigned long Timestamp;
TD005_MULTIPLEVALUES MultipleValues;

/*
** read channels triggered by timer
*/
result = tdrv005timerMultipleChannelReadWait( FileDescriptor,
                                              &MultipleValues,
                                              &Timestamp,
                                              &MoreDataAvailable,
                                              TD005_WAIT_FOREVER );
if (result == TEWS_OK)
{
      printf( “Timestamp  = %ld\n”, Timestamp );
      printf( “Channel(0) = 0x%08lX\n”,
                                    MultipleValues.Channel[0].Value );
      printf( “Channel(5) = 0x%08lX\n”,
                                    MultipleValues.Channel[5].Value );
} else {
      /* handle error */
}
```

3.5 Digital Input Functions

3.5.1 tdrv005digitalRead()

NAME

tdrv005digitalRead() – reads the digital input lines.

SYNOPSIS

```
int tdrv005digitalRead
(
      int             FileDescriptor,
      unsigned char   *Data
);
```

DESCRIPTION

Reads the current values of all digital input lines. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Data

This parameter points to an *unsigned char* value where the data register content is stored. Channel 0 is represented by bit 0, channel 5 is represented by bit 5 of the returned byte value.

RETURN VALUE

TEWS_OK if the read operation was successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
unsigned char DigitalValue;

/*
** read the current interval timer value
*/
result = tdrv005digitalRead( FileDescriptor, &DigitalValue);
if (result == TEWS_OK)
{
      printf( “Digital Value = 0x%02X\n”, DigitalValue );
}
```

3.5.2 tdrv005digitalWait()

NAME

tdrv005digitalWait() – waits for an event on a digital input line.

SYNOPSIS

```
int tdrv005digitalWait
(
      int                   FileDescriptor,
      unsigned char         Channel,
      TD005_TRANSITION      Transition,
      int                   Timeout
);
```

DESCRIPTION

Wait for the specified transition (rising or falling edge) on the specified digital input line. The function returns to the caller if the specified transition or the specified timeout occurred.

PARAMETERS

FileDescriptor

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

Channel

This value specifies the channel on which the specified event should occur. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Only one channel may be specified.

Transition

This value specifies the channel on which the specified event should occur.

Value	Description
TD005_TR_RISING_EDGE	Rising edge on digital input
TD005_TR_FALLING_EDGE	Falling edge on digital input

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, TD005_WAIT_FOREVER must be specified.

RETURN VALUE

TEWS_OK if the timer event occurred successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_TIMEOUT	The event has not occurred, timeout.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** wait indefinitely for a rising edge on digital input of channel 1
*/
result = tdrv005digitalWait( FileDescriptor,
                             TD005_CH1,
                             TD005_TR_RISING_EDGE,
                             TD005_WAIT_FOREVER );
if (result < 0)
{
      /* handle error */
}
```

3.6 Global Operation Functions

3.6.1 tdrv005globalChannelEnable()

NAME

tdrv005globalChannelEnable() – globally enables multiple channels.

SYNOPSIS

```
int tdrv005globalChannelEnable
(
      int                  FileDescriptor,
      unsigned char        Channels
);
```

DESCRIPTION

Enable multiple channels (SSI or Counter) simultaneously. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channels

This value specifies the channels which should be enabled. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Multiple channels may be OR'ed to one value.

RETURN VALUE

TEWS_OK if the specified channels were enabled successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** enable channel 0 and channel 5
*/
result = tdrv005globalChannelEnable ( FileDescriptor,
                                      TD005_CH0 | TD005_CH5 );
if (result < 0)
{
      /* handle error */
}
```

3.6.2 tdrv005globalChannelDisable()

NAME

tdrv005globalChannelDisable() – globally disables multiple channels.

SYNOPSIS

```
int tdrv005globalChannelDisable
(
      int             FileDescriptor,
      unsigned char   Channels
);
```

DESCRIPTION

Disable multiple channels (SSI or Counter) simultaneously. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channels

This value specifies the channels which should be disabled. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Multiple channels may be OR'ed to one value.

RETURN VALUE

TEWS_OK if the specified channels were disabled successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** disable channel 0 and channel 5
*/
result = tdrv005globalChannelDisable ( FileDescriptor,
                                        TD005_CH0 | TD005_CH5 );
if (result < 0)
{
      /* handle error */
}
```

3.6.3 tdrv005globalCounterPreloadSet()

NAME

tdrv005globalCounterPreloadSet() – globally sets counter preload registers.

SYNOPSIS

```
int tdrv005globalCounterPreloadSet
(
      int                   FileDescriptor,
      unsigned char         Channels,
      TD005_MULTIPLEVALUES  *MultipleValues
);
```

DESCRIPTION

Perform a simultaneous setup of the preload registers of specified counter channels. The desired values must be supplied to this function as well as the channel numbers which are affected. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channels

This value specifies the channels which should be preloaded. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Multiple channels may be OR'ed to one value.

MultipleValues

This is a pointer to a TD005_MULTIPLEVALUES structure, where the read values are stored. The TD005_MULTIPLEVALUES structure has the following layout:

```
typedef struct
{
        TD005_VALUE_BUF Channel[6];
} TD005_MULTIPLEVALUES;
```

MEMBERS

Channel

This parameter is an array of a TD005_VALUE_BUF structure, which holds the preload data. The value for channel 0 is located at array index 0, channel 5's value is located at array index 5. The TD005_VALUE_BUF structure has the following layout:

```
typedef struct
{
         unsigned long   Value;
         unsigned long   Status;
} TD005_VALUE_BUF;
```

MEMBERS

Value

This parameter holds the preload channel value.

Status

This parameter is not used for this function.

RETURN VALUE

TEWS_OK if the preload registers were set successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Channel is not configured to counter mode.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
TD005_MULTIPLEVALUES Values;

/*
** preload channel 0 and channel 5
*/
Values[0].Value = 0x00000000;
Values[5].Value = 0x50000000;
result = tdrv005globalCounterPreloadSet ( FileDescriptor,
                                          TD005_CH0 | TD005_CH5,
                                          Values );
if (result < 0)
{
      /* handle error */
}
```

3.6.4 tdrv005globalCounterLoad()

NAME

tdrv005globalCounterLoad() – globally loads preload registers into counters.

SYNOPSIS

```
int tdrv005globalCounterLoad
(
      int             FileDescriptor,
      unsigned char   Channels
);
```

DESCRIPTION

Perform a simultaneous preload of specified counter channels. The values stored in the corresponding preload registers are loaded into the specified counters simultaneously. The function returns immediately to the caller.

PARAMETERS

FileDescriptor

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

Channels

This value specifies the channels which should be preloaded. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Multiple channels may be OR'ed to one value.

RETURN VALUE

TEWS_OK if the specified counters were loaded successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified.
TERR_BUSY	Channel is not configured to counter mode.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;

/*
** load channel 0 and channel 5
*/
result = tdrv005globalCounterLoad ( FileDescriptor,
                                    TD005_CH0 | TD005_CH5 );
if (result < 0)
{
      /* handle error */
}
```

3.6.5 tdrv005globalMultipleChannelRead()

NAME

tdrv005globalMultipleChannelRead() – reads multiple channels simultaneously.

SYNOPSIS

```
int tdrv005globalMultipleChannelRead
(
      int                   FileDescriptor,
      unsigned char         Channels,
      TD005_MULTIPLEVALUES  *MultipleValues
);
```

DESCRIPTION

Return the values of simultaneously sampled channels. The desired channel numbers must be supplied to this function as well as an array to store all channel values. The function returns to the caller after the read operation is finished.

Channels configured to SSI listen-only mode may not be used with this function.

PARAMETERS

FileDescriptor

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

Channels

This value specifies the channels which should be read simultaneously. The pre-defined values (TD005_CH0 – TD005_CH5) must be used. Multiple channels may be OR'ed to one value.

MultipleValues

This is a pointer to a TD005_MULTIPLEVALUES structure, where the read values are stored. The returned values are only valid for channels enabled by the parameter *Channels*. The TD005_MULTIPLEVALUES structure has the following layout:

```
typedef struct
{
        TD005_VALUE_BUF Channel[6];
} TD005_MULTIPLEVALUES;
```

MEMBERS

Channel

This parameter is an array of a TD005_VALUE_BUF structure, which holds the returned channel data. The value for channel 0 is returned at array index 0, channel 5's value is located at array index 5. The TD005_VALUE_BUF structure has the following layout:

```
typedef struct
{
         unsigned long   Value;
         unsigned long   Status;
} TD005_VALUE_BUF;
```

MEMBERS

Value

This parameter holds the returned channel value.

Status

This parameter holds the returned channel status.

RETURN VALUE

TEWS_OK if the channel data is read successfully, otherwise a negative error code.

ERRORS

TERR_INVALID_CHANNEL	Invalid channel specified.
TERR_INVALID_PARAMETER	Invalid parameter specified, or a specified channel is not configured properly.
TERR_BUSY	A MultipleChannelRead job is already pending, or a channel is busy with an SSI job.
TERR_TIMEOUT	Internal timeout. The values couldn't be retrieved within 2 seconds.

EXAMPLE

```
#include “tdrv005api.h”
int FileDescriptor;
int result;
TD005_MULTIPLEVALUES MultipleValues;

/*
** read channel 0 and channel 5 simultaneously
*/
result = tdrv005globalMultipleChannelRead ( FileDescriptor,
                                            TD005_CH0 | TD005_CH5,
                                            &MultipleValues );
if (result == TEWS_OK)
{
       printf( “Channel(0) = 0x%08lX\n”,
                                    MultipleValues.Channel[0].Value );
       printf( “Channel(5) = 0x%08lX\n”,
                                    MultipleValues.Channel[5].Value );
} else {
       /* handle error */
}
```

4 Debugging and Diagnostic

If the driver will not work properly, please enable debug outputs by defining the symbols *DEBUG* and *DEBUG_PCI* in file tdrv005.c, and symbols DEBUG, TEWS_DEBUG_ENABLE, DEBUG_INT_ENABLE, DEBUG_IOCTL_ENABLE and DEBUG_ENTRY_ENABLE in file tcdi_lib.h.

The debug output should appear on the console. If not, please check the symbol *KKPF_PORT* in *uparam.h*. This symbol should be configured to a valid COM port (e.g. *SKDB_COM1*).

The debug output displays the device information data for the current major device like this.

```
TDRV005: Start looking for Device 'TPMC117'
Bus = 0  Dev = 11  Func = 0
[00] = 00751498
[04] = 02800003
[08] = 11800000
[0C] = 00000008
[10] = EB021000
[14] = 0000D401
[18] = EB022000
[1C] = 00000000
[20] = 00000000
[24] = 00000000
[28] = 00000000
[2C] = 000A1498
[30] = 00000000
[34] = 00000040
[38] = 00000000
[3C] = 0000010B
TDRV005: Found a compatible module: TPMC117 on BusNo=0, DevNo=11
  MemRes[0]: mapped: 0xE7021000  size=0x80 (128)
  IoRes[0]:  mapped: 0x0000D400  size=0x80 (128)
  MemRes[1]: mapped: 0xE7022000  size=0x100 (256)
```

The debug output above is only an example. Debug output on other systems may be different for addresses and data in some locations.