The Embedded I/O Company

TDRV005-SW-95

QNX6-Neutrino Device Driver

6 Channel SSI, Incremental Encoder, Counter

Version 1.0.x

User Manual

Issue 1.0.0

October 2007

TEWS TECHNOLOGIES GmbH		**TEWS TECHNOLOGIES LLC**	
Am Bahnhof 7	Phone: +49 (0) 4101 4058 0	9190 Double Diamond Parkway,	Phone: +1 (775) 850 5830
25469 Halstenbek, Germany	Fax: +49 (0) 4101 4058 19	Suite 127, Reno, NV 89521, USA	Fax: +1 (775) 201 0347
www.tews.com	e-mail: info@tews.com	www.tews.com	e-mail: usasales@tews.com

TDRV005-SW-95

QNX6-Neutrino Device Driver

6 Channel SSI, Incremental Encoder, Counter

Supported Modules:
TPMC117

Issue	Description	Date
1.0.0	First Issue	October 29, 2007

Table of Contents

1 Introduction

The TDRV005-SW-95 QNX6-Neutrino device driver allows the operation of the TPMC117 PMC conforming to the QNX6-Neutrino I/O system specification.

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-SW-95 device driver and its API supports 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-95 supports the modules listed below:

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

In this document all supported modules and devices will be called TDRV005. Specials for certain devices will be advised.

To get more information about the features and use of TPMC117 devices it is recommended to read the manuals listed below.

TPMC117 User manual

TPMC117 Engineering Manual

2 Installation

Following files are located on the distribution media:

Directory path 'TDRV005-SW-95':

TDRV005-SW-95-SRC.tar.gz	GZIP compressed archive with driver source code
TDRV005-SW-95-1.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

For installation the files have to be copied to the desired target directory.

The GZIP compressed archive TDRV005-SW-95-SRC.tar.gz contains the following files and directories:

Directory path './tdrv005/':

driver/tdrv005.c	TDRV005 device driver source
driver/tdrv005def.h	TDRV005 driver include file
driver/tdrv005.h	TDRV005 include file for driver and application
driver/node.c	Linked-List management source file
driver/node.h	Linked-List management header file
driver/common.mk	Make parameter for driver
driver/Makefile	Recursive Makefile for driver
driver/nto/Makefile	Recursive Makefile for driver
driver/nto/x86/Makefile	Recursive Makefile for driver
driver/nto/x86/o/Makefile	Recursive Makefile for driver
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)
TCOSI/tdrv005api.h	API include file
TCOSI/tdrv005api.c	API source file
example/tdrv005exa.c	Example application
example/common.mk	Make parameter for example
example/Makefile	Recursive Makefile for example
example/nto/Makefile	Recursive Makefile for example
example/nto/x86/Makefile	Recursive Makefile for example
example/nto/x86/o/Makefile	Recursive Makefile for example
include/tpmodule.h	Driver and kernel independent library header file
include/tpmodule.c	Driver and kernel independent library source file
include/tpxxxhwdep.h	HAL library header file
include/tpxxxhwdep.c	HAL library source file

In order to perform an installation, extract all files of the archive TDRV005-SW-95-SRC.tar.gz to the `/usr/src` directory. The command 'tar -xzvf TDRV005-SW-95-SRC.tar.gz' will extract the files into the local directory.

2.1 Build and install the device driver

Change to the target directory
```
# cd /usr/src/tdrv005
```

Change to the driver subdirectory
```
# cd driver
```

Create and install the driver in the module directory
```
# make install
```

2.2 Start the device driver

Start the device driver process in background mode
```
# tdrv005 &
```

For debugging it may be useful to start the device driver in verbose mode, the driver will print out information and error messages. To start the device driver in verbose mode enter:
```
# tdrv005 –v
```

2.3 Stop device driver execution

The driver can be stopped using the *kill* command. To get the ID of the device driver process the *ps* command can be used.

Do not use the *Ctrl^C* command for device driver termination!

Device driver termination using *Ctrl^C* may cause a freeze of the system, a termination with the kill command is save.

2.4 Build example application

First you have to make the application interface file available, the simplest way is to copy into the example directory.

Change into the example application directory:
```
# cd /usr/src/tdrv005/example
```

Now build the example:
```
# make
```

After build has completed the example can be executed (e.g.):
```
# nto/x86/o/tdrv005exa
```

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 devctl() 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.

The files placed in the *TCOSI* subdirectory must be included to the application project.

3.1 General Functions

3.1.1 td005open()

NAME

td005open() – opens a device.

SYNOPSIS

```
int td005open
(
        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 = td005open(“/tdrv005/0”);
if (FileDescriptor < 0)
{
          /* handle open error */
}
```

3.1.2 td005close()

NAME

td005close() – closes a device.

SYNOPSIS

```
int td005close
(
        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 = td005close( FileDescriptor );
if (result < 0)
{
     /* handle close error */
}
```

3.2 SSI Functions

3.2.1 td005ssiSetup()

Name

td005ssiSetup() – sets up a channel for SSI operation.

Synopsis

```
int td005ssiSetup
(
        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 *td005globalChannelEnable*.

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, or data buffer null.

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 = td005ssiSetup( FileDescriptor, TD005_CH0, &Options );
if (result < 0)
{
     /* handle configuration error */
}
```

3.2.2 td005ssiRead()

Name

td005ssiRead() – reads the value of an SSI channel.

Synopsis

```
int td005ssiRead
(
        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 were the data register content is stored.

Status

This parameter points to an unsigned long value were 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. Data pointer is null.

Example

```
#include “tdrv005api.h”

int                FileDescriptor;
int                result;
unsigned long      ssiValue;
unsigned long      ssiStatus;

/*
** read the current counter value
*/
result = td005ssiRead(  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 td005counterSetup()

Name

td005counterSetup() – sets up a channel for counter operation.

Synopsis

```
int td005counterSetup
(
        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 *td005globalChannelEnable*.

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 = td005counterSetup( FileDescriptor, TD005_CH0, &Options );
if (result < 0)
{
     /* handle configuration error */
}
```

3.3.2 td005counterRead()

Name

td005counterRead() – reads the value of a counter channel.

Synopsis

```
int td005counterRead
(
        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 were the data register content is stored.

Status

This parameter points to an unsigned long value were 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. Data pointer is null.
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 = td005counterRead(  FileDescriptor,
                            TD005_CH0,
                            &CounterValue,
                            &Status );
if (result == TEWS_OK)
{
    printf( Counter Value = 0x%.8lX\n”, CounterValue );
    printf( Status  Value = 0x%.8lX\n”, Status );
}
```

3.3.3 td005counterPreloadSet()

Name

td005counterPreloadSet() – sets the preload register of a counter channel.

Synopsis

```
int td005counterPreloadSet
(
        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, data buffer is null.
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 = td005counterPreloadSet( TD005_CH0, PreloadValue );
if (result < 0)
{
    /* handle error */
}
```

3.3.4 td005counterLoad()

Name

td005counterLoad() – loads the preload register into the counter channel.

Synopsis

```
int td005counterLoad
(
        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, data buffer is null.
TERR_BUSY	Channel is not configured to counter mode.

Example

```
#include  “tdrv005api.h”

int       FileDescriptor;
int       result;

/*
** load counter value
*/
result = td005counterLoad( FileDescriptor, TD005_CH0 );
if (result < 0)
{
     /* handle error */
}
```

3.3.5 td005counterReset()

Name

td005counterReset() – resets the counter channel.

Synopsis

```
int td005counterReset
(
        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, data buffer is null.
TERR_BUSY	Channel is not configured to counter mode.

Example

```
#include  “tdrv005api.h”

int       FileDescriptor;
int       result;

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

3.3.6 td005counterWaitMatch()

Name

td005counterWaitMatch() – waits for a counter-match event.

Synopsis

```
int td005counterWaitMatch
(
        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, data buffer is null.
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 = td005counterWaitMatch(  FileDescriptor,
                                 TD005_CH0,
                                 MatchValue,
                                 TD005_WAIT_FOREVER );
if (result < 0)
{
    /* handle error */
}
```

3.3.7 td005counterWaitControlModeEvent()

Name

td005counterWaitControlModeEvent() – waits for a counter-control-mode event.

Synopsis

```
int td005counterWaitControlModeEvent
(
        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, data buffer is null.
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 = td005counterWaitControlModeEvent( FileDescriptor,
                                           TD005_CH0,
                                           TD005_WAIT_FOREVER );
if (result < 0)
{
    /* handle error */
}
```

3.4 Timer Functions

3.4.1 td005timerSetup()

Name

td005timerSetup() – sets up the timer.

Synopsis

```
int td005timerSetup
(
        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
TP005_CLKFRQ_1MHZ	1 MHz clock
TP005_CLKFRQ_2MHZ	2 MHz clock
TP005_CLKFRQ_4MHZ	4 MHz clock
TP005_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 = td005timerSetup(    FileDescriptor,
                             TD005_CLKFRQ_1MHZ,
                             10000 );
if (result < 0)
{
     /* handle error */
}
```

3.4.2 td005timerStart()

Name

td005timerStart() – starts the timer.

Synopsis

```
int td005timerStart
(
        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 = td005timerStart( FileDescriptor );
if (result < 0)
{
     /* handle error */
}
```

3.4.3 td005timerStop()

Name

td005timerStop() – stops the timer.

Synopsis

```
int td005timerStop
(
        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 = td005timerStop( FileDescriptor );
if (result < 0)
{
    /* handle error */
}
```

3.4.4 td005timerRead()

Name

td005timerRead() – reads the current timer value.

Synopsis

```
int td005timerRead
(
        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, data pointer is null.

Example

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

/*
** read the current interval timer value
*/
result = td005timerRead( FileDescriptor, &TimerValue );
if (result == TEWS_OK)
{
     printf( “Timer Value = 0x%.4X\n”, TimerValue );
}
```

3.4.5 td005timerWait()

Name

td005timerWait() – waits for a timer event.

Synopsis

```
int td005timerWait
(
        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 = td005timerWait( FileDescriptor, TD005_WAIT_FOREVER );
if (result < 0)
{
     /* handle error */
}
```

3.4.6 td005timerMultipleChannelReadSetup()

Name

td005timerMultipleChannelReadSetupt() – sets up channels for multiple read.

Synopsis

```
int td005timerMultipleChannelReadSetup
(
        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, buffer is NULL.
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 = td005timerMultipleChannelReadSetup(   FileDescriptor,
                                               TD005_CH0 | TD005_CH5 );
if (result < 0)
{
     /* handle error */
}
```

3.4.7 td005timerMultipleChannelReadWait()

Name

td005timerMultipleChannelReadWait() – waits for the simultaneous data timer event.

Synopsis

```
int td005timerMultipleChannelReadWait
(
        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, output buffer is NULL.
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 = td005timerMultipleChannelReadWait(    FileDescriptor,
                                               &MultipleValues,
                                               &Timestamp,
                                               &MoreDataAvailable,
                                               TD005_WAIT_FOREVER );
if (result == TEWS_OK)
{
    printf( “Timestamp  = %ld\n”, Timestamp );
    printf( “Channel(0) = 0x%.8lX\n”, MultipleValues.Channel[0].Value );
    printf( “Channel(5) = 0x%.8lX\n”, MultipleValues.Channel[5].Value );
}
else
{
    /* handle error */
}
```

3.5 Digital Input Functions

3.5.1 td005digitalRead()

Name

td005digitalRead() – reads the digital input lines.

Synopsis

```
int td005digitalRead
(
        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 were 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, data pointer is null.

Example

```
#include  “tdrv005api.h”

int            FileDescriptor;
int            result;
unsigned char  DigitalValue;

/*
** read the current interval timer value
*/
result = td005digitalRead( FileDescriptor, &DigitalValue);
if (result == TEWS_OK)
{
     printf( “Digital Value = 0x%.2X\n”, DigitalValue );
}
```

3.5.2 td005digitalWait()

Name

td005digitalWait() – waits for an event on a digital input line.

Synopsis

```
int td005digitalWait
(
        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 = td005digitalWait(   FileDescriptor,
                             TD005_CH1,
                             TD005_TR_RISING_EDGE,
                             TD005_WAIT_FOREVER );
if (result < 0)
{
    /* handle error */
}
```

3.6 Global Operation Functions

3.6.1 td005globalChannelEnable()

Name

td005globalChannelEnable() – globally enables multiple channels.

Synopsis

```
int td005globalChannelEnable
(
        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, data buffer is null.

Example

```
#include  “tdrv005api.h”

int       FileDescriptor;
int       result;

/*
** enable channel 0 and channel 5
*/
result = td005globalChannelEnable (   FileDescriptor,
                                      TD005_CH0 | TD005_CH5 );
if (result < 0)
{
     /* handle error */
}
```

3.6.2 td005globalChannelDisable()

Name

td005globalChannelDisable() – globally disables multiple channels.

Synopsis

```
int td005globalChannelDisable
(
        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, data buffer is null.

Example

```
#include  "tdrv005api.h"

int       FileDescriptor;
int       result;

/*
** disable channel 0 and channel 5
*/
result = td005globalChannelDisable (  FileDescriptor,
                                      TD005_CH0 | TD005_CH5 );
if (result < 0)
{
     /* handle error */
}
```

3.6.3 td005globalCounterPreloadSet()

Name

td005globalCounterPreloadSet() – globally sets counter preload registers.

Synopsis

```
int td005globalCounterPreloadSet
(
        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, data pointer is null.
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 = td005globalCounterPreloadSet (    FileDescriptor,
                                           TD005_CH0 | TD005_CH5,
                                           Values );

if (result < 0)
{
     /* handle error */
}
```

3.6.4 td005globalCounterLoad()

Name

td005globalCounterLoad() – globally loads preload registers into counters.

Synopsis

```
int td005globalCounterLoad
(
        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, data buffer is null.
TERR_BUSY	Channel is not configured to counter mode.

Example

```
#include  “tdrv005api.h”

int       FileDescriptor;
int       result;

/*
** load channel 0 and channel 5
*/
result = td005globalCounterLoad (FileDescriptor,
                                TD005_CH0 | TD005_CH5 );
if (result < 0)
{
    /* handle error */
}
```

3.6.5 td005globalMultipleChannelRead()

Name

td005globalMultipleChannelRead() – reads multiple channels simultaneously.

Synopsis

```
int td005globalMultipleChannelRead
(
        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, data pointer is null, 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 = td005globalMultipleChannelRead (  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 I/O Functions