

TDRV005-SW-65

Windows 2000/XP Device Driver

6 Channel SSI, Incremental Encoder, Counter

Version 1.0.x

User Manual

Issue 1.0.1

June 2008

TEWS TECHNOLOGIES GmbH

Am Bahnhof 7
25469 Halstenbek, Germany
www.tews.com

Phone: +49 (0) 4101 4058 0
Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com

TEWS TECHNOLOGIES LLC

9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com

Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TDRV005-SW-65

Windows 2000/XP Device Driver

6 Channel SSI, Incremental Encoder, Counter

Supported Modules:

TPMC117

This document contains information, which is proprietary to TEWS TECHNOLOGIES GmbH. Any reproduction without written permission is forbidden.

TEWS TECHNOLOGIES GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS TECHNOLOGIES GmbH reserves the right to change the product described in this document at any time without notice.

TEWS TECHNOLOGIES GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2007-2008 by TEWS TECHNOLOGIES GmbH

Issue	Description	Date
1.0.0	First Issue	May 10, 2007
1.0.1	Files moved to subdirectory	June 23, 2008

Table of Contents

1	INTRODUCTION.....	4
2	INSTALLATION.....	5
2.1	Software Installation.....	5
2.1.1	Windows 2000 / XP.....	5
2.1.2	Confirming Windows 2000 / XP Installation.....	6
3	TDRV005 API DOCUMENTATION.....	7
3.1	General Functions.....	7
3.1.1	tdrv005open().....	7
3.1.2	tdrv005close().....	9
3.2	SSI Functions.....	11
3.2.1	tdrv005ssiSetup().....	11
3.2.2	tdrv005ssiRead().....	14
3.3	Counter Functions.....	16
3.3.1	tdrv005counterSetup().....	16
3.3.2	tdrv005counterRead().....	20
3.3.3	tdrv005counterPreloadSet().....	22
3.3.4	tdrv005counterLoad().....	24
3.3.5	tdrv005counterReset().....	26
3.3.6	tdrv005counterWaitMatch().....	28
3.3.7	tdrv005counterWaitControlModeEvent().....	30
3.4	Timer Functions.....	32
3.4.1	tdrv005timerSetup().....	32
3.4.2	tdrv005timerStart().....	34
3.4.3	tdrv005timerStop().....	36
3.4.4	tdrv005timerRead().....	38
3.4.5	tdrv005timerWait().....	40
3.4.6	tdrv005timerMultipleChannelReadSetup().....	42
3.4.7	tdrv005timerMultipleChannelReadWait().....	44
3.5	Digital Input Functions.....	47
3.5.1	tdrv005digitalRead().....	47
3.5.2	tdrv005digitalWait().....	49
3.6	Global Operation Functions.....	51
3.6.1	tdrv005globalChannelEnable().....	51
3.6.2	tdrv005globalChannelDisable().....	53
3.6.3	tdrv005globalCounterPreloadSet().....	55
3.6.4	tdrv005globalCounterLoad().....	58
3.6.5	tdrv005globalMultipleChannelRead().....	60

1 Introduction

The TDRV005-SW-65 Windows WDM device driver allows the operation of the TPMC117 PMC conforming to the Windows WDM I/O system specification. This includes a device-independent basic I/O interface with open(), close() and ioctl() functions.

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-42 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-65 device driver supports the modules listed below:

TPMC117 6 Channel SSI, Incremental Encoder, Counter (PMC)

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 directory TDRV005-SW-65 on the distribution media:

tdrv005.sys	Windows driver binary
tdrv005.h	Header-file with IOCTL code definitions
tdrv005.inf	Windows installation script
TDRV005-SW-65-1.0.1.pdf	This document in PDF-Format
tdrv005api.h	API include file
tdrv005api.c	API source file
TCOSI/tcosi_lib.h	OS function abstraction include file
TCOSI/tcosi_lib.c	OS function abstraction source
\\example\tdrv005exa.c	Microsoft Visual C example application
ChangeLog.txt	Release history
Release.txt	Information about the Device Driver Release

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

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV005 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV005 card(s) and boot-up your system, Windows 2000 / XP setup will show a "**New hardware found**" dialog box.

1. The "**Upgrade Device Driver Wizard**" dialog box will appear on your screen. Click "**Next**" button to continue.
2. In the following dialog box, choose "**Search for a suitable driver for my device**". Click "**Next**" button to continue.
3. Insert the TDRV005 driver media; select the matching media drive in the dialog box. Click "**Next**" button to continue.
4. Now the driver wizard should find a suitable device driver on the media. Click "**Next**" button to continue.
5. Complete the upgrade device driver and click "**Finish**" to take all the changes effect.
6. Now copy all needed files (tdrv005.h, tdrv005api.h, tdrv005api.c, TDRV005-SW-65-x.x.x.pdf, ...) to the desired target directories.

After successful installation the TDRV005 device driver will start immediately and creates devices (TDRV005_1, TDRV005_2 ...) for all recognized TDRV005 modules.

2.1.2 Confirming Windows 2000 / XP Installation

To confirm that the driver has been properly loaded in Windows 2000 / XP, perform the following steps:

1. From Windows 2000 / XP, open the "**Control Panel**" from "**My Computer**".
2. Click the "**System**" icon and choose the "**Hardware**" tab, and then click the "**Device Manager**" button.
3. Click the "+" in front of "**Other Devices**".
The driver "**TEWS TECHNOLOGIES TDRV005 (6 Channel SSI, Incr. Enc., Counter)**" should appear.

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

```
TEWS_HANDLE 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

<code>TERR_INVALID_HANDLE_VALUE</code>	The specified device does not exist.
--	--------------------------------------

Example

```
#include "tdrv005api.h"
TEWS_HANDLE FileDescriptor;

/*
** open file descriptor to device
*/
FileDescriptor = tdrv005open( "/tdrv005_0" );
if (FileDescriptor < 0)
{
    /* handle open error */
}
```


3.1.2 tdrv005close()

Name

tdrv005close() – closes a device.

Synopsis

```
int tdrv005close
(
    TEWS_HANDLE          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"
TEWS_HANDLE 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
(
    TEWS_HANDLE          FileDescriptor,
    unsigned char        Channel,
    TDRV005_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 (TDRV005_CH0 – TDRV005_CH5) must be used. Only one channel may be specified.

Options

This value specifies the necessary configuration options in the structure *TDRV005_SSI_SETUP* with the following layout:

```
typedef struct
{
    TDRV005_SSI_MODE           Mode;
    unsigned char              NumberOfDataBits;
    TDRV005_SSI_CODING        Coding;
    unsigned char              ZeroBits;
    TDRV005_SSI_PARITY        Parity;
    unsigned char              ClockRate;
} TDRV005_SSI_SETUP;
```

Members

Mode

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

Value	Description
TDRV005_MODE_STANDARD	Standard SSI operation
TDRV005_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
TDRV005_CODING_BINARY	Binary coding is used.
TDRV005_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
TDRV005_PARITY_NONE	No parity bit is used.
TDRV005_PARITY_EVEN	An even parity bit is used.
TDRV005_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"

TEWS_HANDLE      FileDescriptor;
int              result;
TDRV005_SSI_SETUP Options;

/*
** setup the counter with appropriate options
*/
Options.Mode                = TDRV005_MODE_STANDARD;
Options.NumberOfDataBits    = 32;
Options.Coding              = TDRV005_CODING_BINARY;
Options.ZeroBits           = 1;
Options.Parity              = TDRV005_PARITY_NONE;
Options.ClockRate          = 10;

result = tdrv005ssiSetup( FileDescriptor, TDRV005_CH0, &Options );
if (result < 0)
{
    /* handle configuration error */
}
```

3.2.2 tdrv005ssiRead()

Name

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

Synopsis

```
int tdrv005ssiRead
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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, *TDRV005_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. Data pointer is null.

Example

```
#include "tdrv005api.h"

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

/*
** read the current counter value
*/
result = tdrv005ssiRead( FileDescriptor,
                        TDRV005_CH0,
                        TDRV005_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
(
    TEWS_HANDLE          FileDescriptor,
    unsigned char        Channel,
    TDRV005_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 (TDRV005_CH0 – TDRV005_CH5) must be used. Only one channel may be specified.

Options

This value specifies the necessary configuration options in the structure `TDRV005_COUNTER_SETUP` with the following layout:

```
typedef struct
{
    unsigned char          Polarity;
    TDRV005_CNT_INPUT      InputMode;
    TDRV005_CNT_INDEX      IndexControlMode;
    TDRV005_CNT_SCM        SpecialCountMode;
    TDRV005_CNT_CLKDIV     ClockPrescaler;
} TDRV005_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
TDRV005_POLARITY_A_LOW	low-active signal, default is high-active
TDRV005_POLARITY_B_LOW	low-active signal, default is high-active
TDRV005_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
TDRV005_INPUT_TIMER_UP	Timer mode Up	internal clock prescaler
TDRV005_INPUT_TIMER_DOWN	Timer mode Down	internal clock prescaler
TDRV005_INPUT_DIRECTION	Direction count	Input A & Input B
TDRV005_INPUT_UPDOWN	Up/Down count	Input A & Input B
TDRV005_INPUT_QUADRATURE_1X	Quadrature count 1x	Input A & Input B
TDRV005_INPUT_QUADRATURE_2X	Quadrature count 2x	Input A & Input B
TDRV005_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
TDRV005_ICM_NO_INDEX_CONTROL	no I-control
TDRV005_ICM_LOAD_ON_INDEX	load on index signal
TDRV005_ICM_LATCH_ON_INDEX	latch on index signal
TDRV005_ICM_GATE_ON_INDEX	gate on index signal
TDRV005_ICM_RESET_ON_INDEX	reset on index signal
TDRV005_ICM_REFERENCE_MODE	reference mode (quadrature input mode only)
TDRV005_ICM_AUTO_REFERENCE_MODE	auto-reference mode (quadrature input mode only)
TDRV005_ICM_INDEX_MODE	index mode (quadrature input mode only)

SpecialCountMode

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

Value	Description
TDRV005_SCM_CYCLING_COUNTER	No special count mode, cycling counter
TDRV005_SCM_DIVIDE_BY_N	Divide-by-N mode
TDRV005_SCM_SINGLE_CYCLE	Single cycle mode

ClockPrescaler

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

Value	Description
TDRV005_CLKDIV_1X	Prescaler 1x, 32 MHz clock
TDRV005_CLKDIV_2X	Prescaler 2x, 16 MHz clock
TDRV005_CLKDIV_4X	Prescaler 4x, 8 MHz clock
TDRV005_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"

TEWS_HANDLE          FileDescriptor;
int                  result;
TDRV005_COUNTER_SETUP Options;

/*
** setup the counter with appropriate options
*/
Options.Polarity          = TDRV005_POLARITY_A_LOW |
                           TDRV005_POLARITY_B_LOW |
                           TDRV005_POLARITY_I_LOW;
Options.InputMode         = TDRV005_INPUT_UPDOWN;
Options.IndexControlMode  = TDRV005_ICM_NO_INDEX_CONTROL;
Options.SpecialCountMode  = TDRV005_SCM_CYCLING_COUNTER;
Options.ClockPrescaler    = TDRV005_CLKDIV_8X;

result = tdrv005counterSetup( FileDescriptor,
                              TDRV005_CH0,
                              &Options );

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

3.3.2 tdrv005counterRead()

Name

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

Synopsis

```
int tdrv005counterRead
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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. Data pointer is null.
TERR_BUSY	Channel is not configured to counter mode.

Example

```
#include "tdrv005api.h"

TEWS_HANDLE   FileDescriptor;
int           result;
unsigned long CounterValue, Status;

/*
** read the current counter value
*/
result = tdrv005counterRead( FileDescriptor,
                             TDRV005_CH0,
                             &CounterValue,
                             &Status );

if (result == TEWS_OK)
{
    printf( Counter Value = 0x%.8lX\n", CounterValue );
    printf( Status Value = 0x%.8lX\n", Status );
}
```

3.3.3 tdrv005counterPreloadSet()

Name

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

Synopsis

```
int tdrv005counterPreloadSet
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE   FileDescriptor;
int           result;
unsigned long PreloadValue;

/*
** load counter value
*/
PreloadValue = 0x12345678;
result = tdrv005counterPreloadSet( TDRV005_CH0, PreloadValue );
if (result < 0)
{
    /* handle error */
}
```

3.3.4 tdrv005counterLoad()

Name

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

Synopsis

```
int tdrv005counterLoad
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE  FileDescriptor;
int          result;

/*
** load counter value
*/
result = tdrv005counterLoad( FileDescriptor, TDRV005_CH0 );
if (result < 0)
{
    /* handle error */
}
```

3.3.5 tdrv005counterReset()

Name

tdrv005counterReset() – resets the counter channel.

Synopsis

```
int tdrv005counterReset
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE  FileDescriptor;
int          result;

/*
** reset counter value
*/
result = counterReset( FileDescriptor, TDRV005_CH0 );
if (result < 0)
{
    /* handle error */
}
```

3.3.6 tdrv005counterWaitMatch()

Name

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

Synopsis

```
int tdrv005counterWaitMatch
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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, TDRV005_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"

TEWS_HANDLE FileDescriptor;
int result;
unsigned long MatchValue;

/*
** wait indefinitely for counter match event
*/
MatchValue = 0x12345678;
result = tdrv005counterWaitMatch( FileDescriptor,
                                  TDRV005_CH0,
                                  MatchValue,
                                  TDRV005_WAIT_FOREVER );

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

3.3.7 tdrv005counterWaitControlModeEvent()

Name

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

Synopsis

```
int tdrv005counterWaitControlModeEvent
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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, TDRV005_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"

TEWS_HANDLE FileDescriptor;
int result;

/*
** wait indefinitely for counter control mode event
*/
result = tdrv005counterWaitControlModeEvent( FileDescriptor,
                                              TDRV005_CH0,
                                              TDRV005_WAIT_FOREVER );

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

3.4 Timer Functions

3.4.1 tdrv005timerSetup()

Name

tdrv005timerSetup() – sets up the timer.

Synopsis

```
int tdrv005timerSetup
(
    TEWS_HANDLE           FileDescriptor,
    TDRV005_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"

TEWS_HANDLE    FileDescriptor;
int            result;

/*
** setup the interval timer with an interrupt frequency of 100 Hz
*/
result = tdrv005timerSetup( FileDescriptor,
                            TDRV005_CLKFRQ_1MHZ,
                            10000 );

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

3.4.2 tdrv005timerStart()

Name

tdrv005timerStart() – starts the timer.

Synopsis

```
int tdrv005timerStart  
(  
    TEWS_HANDLE          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"

TEWS_HANDLE  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
(
    TEWS_HANDLE          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"

TEWS_HANDLE   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(  
    TEWS_HANDLE          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"

TEWS_HANDLE      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
(
    TEWS_HANDLE      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, TDRV005_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"

TEWS_HANDLE   FileDescriptor;
int           result;

/*
** wait indefinitely for an interval timer event
*/
result = tdrv005timerWait( FileDescriptor, TDRV005_WAIT_FOREVER );
if (result < 0)
{
    /* handle error */
}
```

3.4.6 tdrv005timerMultipleChannelReadSetup()

Name

tdrv005timerMultipleChannelReadSetup() – sets up channels for multiple read.

Synopsis

```
int tdrv005timerMultipleChannelReadSetup
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE FileDescriptor;
int result;

/*
** setup channels 0+5 for simultaneous sampling triggered by timer
*/
result = tdrv005timerMultipleChannelReadSetup( FileDescriptor,
                                               TDRV005_CH0 | TDRV005_CH5 );

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

3.4.7 tdrv005timerMultipleChannelReadWait()

Name

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

Synopsis

```
int tdrv005timerMultipleChannelReadWait
(
    TEWS_HANDLE           FileDescriptor,
    TDRV005_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 TDRV005_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 TDRV005_MULTIPLEVALUES structure has the following layout:

```
typedef struct
{
    TDRV005_VALUE_BUF Channel[6];
} TDRV005_MULTIPLEVALUES;
```

Members

Channel

This parameter is an array of a TDRV005_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 TDRV005_VALUE_BUF structure has the following layout:

```
typedef struct
{
    unsigned long    Value;
    unsigned long    Status;
} TDRV005_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 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 `tdrv005timerMultipleChannelReadWait` must be performed.

Timeout

This value specifies the timeout in milliseconds. If the function should wait indefinitely for the event to occur, `TDRV005_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"

TEWS_HANDLE      FileDescriptor;
int              result;
int              MoreDataAvailable;
unsigned long     Timestamp;
TDRV005_MULTIPLEVALUES MultipleValues;

/*
** read channels triggered by timer
*/
result = tdrv005timerMultipleChannelReadWait( FileDescriptor,
                                              &MultipleValues,
                                              &Timestamp,
                                              &MoreDataAvailable,
                                              TDRV005_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 tdrv005digitalRead()

Name

tdrv005digitalRead() – reads the digital input lines.

Synopsis

```
int tdrv005digitalRead
(
    TEWS_HANDLE          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, data pointer is null.

Example

```
#include "tdrv005api.h"

TEWS_HANDLE  FileDescriptor;
int          result;
unsigned char DigitalValue;

/*
** read the current interval timer value
*/
result = tdrv005digitalRead( FileDescriptor, &DigitalValue);
if (result == TEWS_OK)
{
    printf( "Digital Value = 0x%.2X\n", DigitalValue );
}
```


3.5.2 tdrv005digitalWait()

Name

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

Synopsis

```
int tdrv005digitalWait
(
    TEWS_HANDLE           FileDescriptor,
    unsigned char         Channel,
    TDRV005_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 (TDRV005_CH0 – TDRV005_CH5) must be used. Only one channel may be specified.

Transition

This value specifies the event to wait for. The following values are possible:

Value	Description
TDRV005_TR_RISING_EDGE	Rising edge on digital input
TDRV005_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, TDRV005_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"

TEWS_HANDLE FileDescriptor;
int result;

/*
** wait indefinitely for a rising edge on digital input of channel 1
*/
result = tdrv005digitalWait( FileDescriptor,
                             TDRV005_CH1,
                             TDRV005_TR_RISING_EDGE,
                             TDRV005_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
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE  FileDescriptor;
int          result;

/*
** enable channel 0 and channel 5
*/
result = tdrv005globalChannelEnable ( FileDescriptor,
                                     TDRV005_CH0 | TDRV005_CH5 );

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

3.6.2 tdrv005globalChannelDisable()

Name

tdrv005globalChannelDisable() – globally disables multiple channels.

Synopsis

```
int tdrv005globalChannelDisable
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE FileDescriptor;
int result;

/*
** disable channel 0 and channel 5
*/
result = tdrv005globalChannelDisable( FileDescriptor,
                                     TDRV005_CH0 | TDRV005_CH5 );

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

3.6.3 tdrv005globalCounterPreloadSet()

Name

tdrv005globalCounterPreloadSet() – globally sets counter preload registers.

Synopsis

```
int tdrv005globalCounterPreloadSet
(
    TEWS_HANDLE          FileDescriptor,
    unsigned char        Channels,
    TDRV005_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 (TDRV005_CH0 – TDRV005_CH5) must be used. Multiple channels may be OR'ed to one value.

MultipleValues

This is a pointer to a TDRV005_MULTIPLEVALUES structure, where the read values are stored. The TDRV005_MULTIPLEVALUES structure has the following layout:

```
typedef struct
{
    TDRV005_VALUE_BUF Channel[6];
} TDRV005_MULTIPLEVALUES;
```

Members

Channel

This parameter is an array of a TDRV005_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 TDRV005_VALUE_BUF structure has the following layout:

```
typedef struct
{
    unsigned long    Value;
    unsigned long    Status;
} TDRV005_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"

TEWS_HANDLE          FileDescriptor;
int                  result;
TDRV005_MULTIPLEVALUES Values;

/*
** preload channel 0 and channel 5
*/
Values[0].Value = 0x00000000;
Values[5].Value = 0x50000000;
result = tdrv005globalCounterPreloadSet ( FileDescriptor,
                                           TDRV005_CH0 | TDRV005_CH5,
                                           Values );

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

3.6.4 tdrv005globalCounterLoad()

Name

tdrv005globalCounterLoad() – globally loads preload registers into counters.

Synopsis

```
int tdrv005globalCounterLoad
(
    TEWS_HANDLE          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 (TDRV005_CH0 – TDRV005_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"

TEWS_HANDLE   FileDescriptor;
int           result;

/*
** load channel 0 and channel 5
*/
result = tdrv005globalCounterLoad ( FileDescriptor,
                                     TDRV005_CH0 | TDRV005_CH5 );

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

3.6.5 tdrv005globalMultipleChannelRead()

Name

tdrv005globalMultipleChannelRead() – reads multiple channels simultaneously.

Synopsis

```
int tdrv005globalMultipleChannelRead
(
    TEWS_HANDLE           FileDescriptor,
    unsigned char         Channels,
    TDRV005_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 (TDRV005_CH0 – TDRV005_CH5) must be used. Multiple channels may be OR'ed to one value.

MultipleValues

This is a pointer to a TDRV005_MULTIPLEVALUES structure, where the read values are stored. The returned values are only valid for channels enabled by the parameter *Channels*. The TDRV005_MULTIPLEVALUES structure has the following layout:

```
typedef struct
{
    TDRV005_VALUE_BUF Channel[6];
} TDRV005_MULTIPLEVALUES;
```

Members

Channel

This parameter is an array of a TDRV005_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 TDRV005_VALUE_BUF structure has the following layout:

```
typedef struct
{
    unsigned long    Value;
    unsigned long    Status;
} TDRV005_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"

TEWS_HANDLE      FileDescriptor;
int              result;
TDRV005_MULTIPLEVALUES MultipleValues;

/*
** read channel 0 and channel 5 simultaneously
*/
result = tdrv005globalMultipleChannelRead ( FileDescriptor,
                                             TDRV005_CH0 | TDRV005_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 */
}
```