The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV005-SW-65

Windows Device Driver

6 Channel SSI, Incremental Encoder, Counter

Version 2.0.x

User Manual

Issue 2.0.0

August 2011

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

TDRV005-SW-65

Windows Device Driver

6 Channel SSI, Incremental Encoder, Counter

Supported Modules:
TPMC117
TPMC317

Issue	Description	Date
1.0.0	First Issue	May 10, 2007
1.0.1	Files moved to subdirectory	June 23, 2008
2.0.0	Windows 7 support added, Address TEWS LLC removed, Support of TPMC317 added, General Revision	August 10, 2011

Table of Contents

1 Introduction

The TDRV005-SW-65 Windows device driver is a kernel mode driver which allows the operation of the supported hardware modules on an Intel or Intel-compatible Windows operating system. Supported Windows versions are:

- Windows 2000
- Windows XP
- Windows XP Embedded
- Windows 7 (32bit and 64bit)

The TDRV005-SW-42 device driver 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-65 device driver supports the modules listed below:

TPMC117	6 Channel SSI, Incremental Encoder, Counter	(PMC)
TPMC317	6 Channel SSI, Incremental Encoder, Counter (Conduction Cooled)	(PMC)

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

TPMC117, TPMC317 User Manual
TPMC117, TPMC317 Engineering Manual

2 Installation

Following files are located in directory TDRV005-SW-65 on the distribution media:

i386\	Directory containing driver files for 32bit Windows versions
amd64\	Directory containing driver files for 64bit Windows versions
installer_32bit.exe	Installation tool for 32bit systems (Windows XP or later)
installer_64bit.exe	Installation tool for 64bit systems (Windows XP or later)
tdrv005.h	Header-file with IOCTL code definitions and structure definitions
tdrv005.inf	Windows installation script
api\tdrv005api.h	API include file
api\tdrv005api.c	API source file
example\tdrv005exa.c	Microsoft Visual C example application
TDRV005-SW-65-2.0.0.pdf	This document in PDF-Format
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

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 hardware 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. In Drive A, insert the TDRV005 driver disk; select "***Disk Drive***" in the dialog box.
   Click "***Next***" button to continue.
4. Now the driver wizard should find a suitable device driver on the diskette.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Repeat the steps above for each found module of the TDRV005 product family.
7. Copy needed files (tdrv005.h, API files) to desired target directory.

After successful installation a device is created for each module found (TDRV005_1, TDRV005_2, ...).

2.1.2 Windows 7

This section describes how to install the TDRV005-SW-65 Device Driver on a Windows 7 (32bit or 64bit) operating system.

Depending on the operating system type, execute the installer binaries for either 32bit or 64bit systems. This will install all required driver files using an installation wizard.

Copy needed files (tdrv005.h, API files) to desired target directory.

After successful installation a device is created for each module found (TDRV005_1, TDRV005_2 ...).

2.2 Confirming Driver Installation

To confirm that the driver has been properly loaded, perform the following steps:

1. Open the Windows Device Manager:

   a. For Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***" and click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.

   b. For Windows 7, open the "***Control Panel***" from "***My Computer***" and then click the "***Device Manager***" entry.

2. Click the "**+**" in front of "***Embedded I/O***".
   The driver "***TEWS TECHNOLOGIES - TDRV005 (6 Channel SSI, Incr. Enc., Counter)***" should appear for each installed device.

3 API Documentation

3.1 General Functions

3.1.1 tdrv005open

NAME

tdrv005open() – open a device.

SYNOPSIS

```
TDRV005_HANDLE tdrv005open
(
      char          *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;

/*
** open file descriptor to device
*/
hdl = tdrv005open(“tdrv005_1”);
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. To get extended error information, call ***GetLastError***.

ERROR CODES

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

3.1.2 tdrv005close

NAME

tdrv005close() – closes a device.

SYNOPSIS

```
TDRV005_STATUS tdrv005close
(
    TDRV005_HANDLE          hdl
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** close file descriptor to device
*/
result = tdrv005close( hdl );
if (result != TDRV005_OK)
{
    /* handle close error */
}
```

RETURNS

TDRV005_OK if the device was closed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.

3.2 SSI Functions

3.2.1 tdrv005ssiSetup

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005ssiSetup
(
    TDRV005_HANDLE      hdl
    UCHAR               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

hdl

This value specifies the device handle 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;
        UCHAR                   NumberOfDataBits;
        TDRV005_SSI_CODING      Coding;
        UCHAR                   ZeroBits;
        TDRV005_SSI_PARITY      Parity;
        UCHAR                   ClockRate;
} TDRV005_SSI_SETUP;
```

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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     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( hld, TDRV005_CH0, &Options );
if (result != TDRV005_OK)
{
     /* handle configuration error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified.

3.2.2 tdrv005ssiRead

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005ssiRead
(
    TDRV005_HANDLE      hdl
    UCHAR               Channel,
    int                 Timeout,
    ULONG               *Data,
    ULONG               *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

hdl

This value specifies the device handle 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 ULONG value were the data register content is stored.

Status

This parameter points to an ULONG value were the status register content is stored.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
ULONG              ssiValue;
ULONG              ssiStatus;

/*
** read the current counter value
*/
result = tdrv005ssiRead( hdl,
                         TDRV005_CH0,
                         TDRV005_WAIT_FOREVER,
                         &ssiValue,
                         &ssiStatus);
if (result == TDRV005_OK)
{
     printf( SSI Value  = 0x%08lX\n”, ssiValue );
     printf( SSI Status = 0x%08lX\n”, ssiStatus );
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified or data pointer is NULL.
TDRV005_ERR_BUSY	There is another SSI job in progress.
TDRV005_ERR_NOT_RUNNING	Channel is not configured to SSI mode.

3.3 Counter Functions

3.3.1 tdrv005counterSetup

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005counterSetup
(
    TDRV005_HANDLE          hdl
    UCHAR                   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

hdl

This value specifies the device handle 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
{
    UCHAR                   Polarity;
    TDRV005_CNT_INPUT       InputMode;
    TDRV005_CNT_INDEX       IndexControlMode;
    TDRV005_CNT_SCM         SpecialCountMode;
    TDRV005_CNT_CLKDIV      ClockPrescaler;
} TDRV005_COUNTER_SETUP;
```

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

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE          hdl;
TDRV005_STATUS          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( hdl,
                              TDRV005_CH0,
                              &Options );
if (result != TDRV005_OK)
{
     /* handle configuration error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified.

3.3.2 tdrv005counterRead

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005counterRead
(
    TDRV005_HANDLE      hdl
    UCHAR               Channel,
    ULONG               *Data,
    ULONG               *Status
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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 ULONG value were the data register content is stored.

Status

This parameter points to an ULONG value were the status register content is stored.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
ULONG              CounterValue, Status;

/*
** read the current counter value
*/
result = tdrv005counterRead( hdl,
                             TDRV005_CH0,
                             &CounterValue,
                             &Status );
if (result == TDRV005_OK)
{
     printf( Counter Value = 0x%.8lX\n”, CounterValue );
     printf( Status  Value = 0x%.8lX\n”, Status );
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified. Data pointer is null.
TDRV005_ERR_BUSY	Channel is not configured to counter mode.

3.3.3 tdrv005counterPreloadSet

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005counterPreloadSet
(
    TDRV005_HANDLE      hdl
    UCHAR               Channel,
    ULONG               PreloadValue
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
ULONG              PreloadValue;

/*
** load counter value
*/
PreloadValue = 0x12345678;
result = tdrv005counterPreloadSet(hdl, TDRV005_CH0, PreloadValue );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified. Data pointer is null.
TDRV005_ERR_BUSY	Channel is not configured to counter mode.

3.3.4 tdrv005counterLoad

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005counterLoad
(
    TDRV005_HANDLE      hdl
    UCHAR               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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** load counter value
*/
result = tdrv005counterLoad( hdl, TDRV005_CH0 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified. Data pointer is null.
TDRV005_ERR_BUSY	Channel is not configured to counter mode.

3.3.5 tdrv005counterReset

NAME

tdrv005counterReset() – resets the counter channel.

SYNOPSIS

```
TDRV005_STATUS tdrv005counterReset
(
    TDRV005_HANDLE      hdl
    UCHAR               Channel
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** reset counter value
*/
result = tdrv005counterReset( hdl, TDRV005_CH0 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified. Data pointer is null.
TDRV005_ERR_BUSY	Channel is not configured to counter mode.

3.3.6 tdrv005counterWaitMatch

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005counterWaitMatch
(
    TDRV005_HANDLE      hdl
    UCHAR               Channel,
    ULONG               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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
ULONG              MatchValue;

/*
** wait indefinitely for counter match event
*/
MatchValue = 0x12345678;
result = tdrv005counterWaitMatch(hdl,
                                 TDRV005_CH0,
                                 MatchValue,
                                 TDRV005_WAIT_FOREVER );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified. Data pointer is NULL.
TDRV005_ERR_TIMEOUT	The event has not occurred, timeout.
TDRV005_ERR_BUSY	There is another counter match job in progress.
TDRV005_ERR_NOT_RUNNING	Channel is not configured to counter mode.

3.3.7 tdrv005counterWaitControlModeEvent

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005counterWaitControlModeEvent
(
    TDRV005_HANDLE      hdl
    UCHAR               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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
int                result;

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

if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified. Data pointer is NULL.
TDRV005_ERR_TIMEOUT	The event has not occurred, timeout.
TDRV005_ERR_BUSY	There is another counter control job in progress.
TDRV005_ERR_NOT_RUNNING	Channel is not configured to counter mode.

3.4 Timer Functions

3.4.1 tdrv005timerSetup

NAME

tdrv005timerSetup() – sets up the timer.

SYNOPSIS

```
TDRV005_STATUS tdrv005timerSetup
(
    TDRV005_HANDLE          hdl
    TDRV005_CNT_CLKFRQ      ClockFrequency,
    USHORT                  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

hdl

This value specifies the device handle 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
TDRV005_CLKFRQ_1MHZ	1 MHz clock
TDRV 005_CLKFRQ_2MHZ	2 MHz clock
TDRV 005_CLKFRQ_4MHZ	4 MHz clock
TDRV 005_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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** setup the interval timer with an interrupt frequency of 100 Hz
*/
result = tdrv005timerSetup(  hdl,
                             TDRV005_CLKFRQ_1MHZ,
                             10000 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_INVAL	Invalid parameter specified.

3.4.2 tdrv005timerStart

NAME

tdrv005timerStart() – starts the timer.

SYNOPSIS

```
TDRV005_STATUS tdrv005timerStart
(
    TDRV005_HANDLE          hdl
)
```

DESCRIPTION

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

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** start the previously configured interval timer
*/
result = tdrv005timerStart( hdl );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_NO_CONFIGURATION	The timer has not been configured properly.

3.4.3 tdrv005timerStop

NAME

tdrv005timerStop() – stops the timer.

SYNOPSIS

```
TDRV005_STATUS tdrv005timerStop
(
     TDRV005_HANDLE              hdl
)
```

DESCRIPTION

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

PARAMETERS

hdl

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

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** stop the interval timer
*/
result = tdrv005timerStop( hdl );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_NO_CONFIGURATION	The timer has not been configured properly.

3.4.4 tdrv005timerRead

NAME

tdrv005timerRead() – reads the current timer value.

SYNOPSIS

```
TDRV005_STATUS tdrv005timerRead
(
    TDRV005_HANDLE      hdl
    USHORT              *Data
)
```

DESCRIPTION

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

PARAMETERS

hdl

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

Data

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

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
USHORT             TimerValue;

/*
** read the current interval timer value
*/
result = tdrv005timerRead( hdl, &TimerValue );
if (result == TDRV005_OK)
{
     printf( Timer Value = 0x%.4X\n”, TimerValue );
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_INVAL	Data pointer is NULL.

3.4.5 tdrv005timerWait

NAME

tdrv005timerWait() – waits for a timer event.

SYNOPSIS

```
TDRV005_STATUS tdrv005timerWait
(
    TDRV005_HANDLE      hdl
    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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** wait indefinitely for an interval timer event
*/
result = tdrv005timerWait( hdl, TDRV005_WAIT_FOREVER );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_NOT_RUNNING	The timer is not running.
TDRV005_ERR_NO_CONFIGURATION	The timer has not been configured properly.
TDRV005_ERR_TIMEOUT	The timer event has not occurred, timeout.
TDRV005_ERR_BUSY	There is another timer job in progress.

3.4.6 tdrv005timerMultipleChannelReadSetup

NAME

tdrv005timerMultipleChannelReadSetupt() – sets up channels for multiple read.

SYNOPSIS

```
TDRV005_STATUS tdrv005timerMultipleChannelReadSetup
(
    TDRV005_HANDLE      hdl
    UCHAR               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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** setup channels 0+5 for simultaneous sampling triggered by timer
*/
result = tdrv005timerMultipleChannelReadSetup( hdl,
                                                TDRV005_CH0 | TDRV005_CH5 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified, or a specified channel is not configured properly.
TDRV005_ERR_INVAL	Invalid parameter, buffer is NULL.
TDRV005_ERR_BUSY	There is another multi channel read job in progress.

3.4.7 tdrv005timerMultipleChannelReadWait

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005timerMultipleChannelReadWait
(
    TDRV005_HANDLE              hdl
    TDRV005_MULTIPLEVALUES      *MultipleValues,
    ULONG                       *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

hdl

This value specifies the device handle 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;
```

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
{
    ULONG       Value;
    ULONG       Status;
} TDRV005_VALUE_BUF;
```

Value

This parameter holds the returned channel value.

Status

This parameter holds the returned channel status.

Timestamp

This is a pointer to an ULONG 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE          hdl;
TDRV005_STATUS          result;
int                     MoreDataAvailable;
ULONG                   Timestamp;
TDRV005_MULTIPLEVALUES  MultipleValues;

/*
** read channels triggered by timer
*/
result = tdrv005timerMultipleChannelReadWait( hdl,
                                              &MultipleValues,
                                              &Timestamp,
                                              &MoreDataAvailable,
                                              TDRV005_WAIT_FOREVER );
if (result == TDRV005_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 */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_NOT_RUNNING	The timer is not running.
TDRV005_ERR_NO_CONFIGURATION	The timer has not been configured properly.
TDRV005_ERR_INVAL	Invalid parameter, buffer is NULL.
TDRV005_ERR_BUSY	There is another multi channel read job in progress.
TDRV005_ERR_TIMEOUT	The timer event has not occurred, timeout.

3.5 Digital Input Functions

3.5.1 tdrv005digitalRead

NAME

tdrv005digitalRead() – reads the digital input lines.

SYNOPSIS

```
TDRV005_STATUS tdrv005digitalRead
(
    TDRV005_HANDLE      hdl
    UCHAR               *Data
)
```

DESCRIPTION

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

PARAMETERS

hdl

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

Data

This parameter points to an *UCHAR* 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
UCHAR              DigitalValue;

/*
** read the current interval timer value
*/
result = tdrv005digitalRead( hdl, &DigitalValue);
if (result == TDRV005_OK)
{
     printf( “Digital Value = 0x%.2X\n”, DigitalValue );
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_INVAL	Data pointer is NULL.

3.5.2 tdrv005digitalWait

NAME

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

SYNOPSIS

```
TDRV005_STATUS tdrv005digitalWait
(
    TDRV005_HANDLE          hdl
    UCHAR                   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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** wait indefinitely for a rising edge on digital input of channel 1
*/
result = tdrv005digitalWait( hdl,
                             TDRV005_CH1,
                             TDRV005_TR_RISING_EDGE,
                             TDRV005_WAIT_FOREVER );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_INVAL	Data pointer is NULL.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_BUSY	There is another digital job in progress.
TDRV005_ERR_TIMEOUT	The specified event has not occurred, timeout.

3.6 Global Operation Functions

3.6.1 tdrv005globalChannelEnable

NAME

tdrv005globalChannelEnable() – globally enables multiple channels.

SYNOPSIS

```
TDRV005_STATUS tdrv005globalChannelEnable
(
    TDRV005_HANDLE      hdl
    UCHAR               Channels
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** enable channel 0 and channel 5
*/
result = tdrv005globalChannelEnable(  hdl,
                                      TDRV005_CH0 | TDRV005_CH5 );
if (result != TDRV005_OK)
{
          /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel(s) specified.

3.6.2 tdrv005globalChannelDisable

NAME

tdrv005globalChannelDisable() – globally disables multiple channels.

SYNOPSIS

```
TDRV005_STATUS tdrv005globalChannelDisable
(
    TDRV005_HANDLE      hdl
    UCHAR               Channels
)
```

DESCRIPTION

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

PARAMETERS

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** disable channel 0 and channel 5
*/
result = tdrv005globalChannelDisable( hdl,
                                      TDRV005_CH0 | TDRV005_CH5 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel(s) specified.

3.6.3 tdrv005globalCounterPreloadSet

NAME

tdrv005globalCounterPreloadSet() – globally sets counter preload registers.

SYNOPSIS

```
TDRV005_STATUS tdrv005globalCounterPreloadSet
(
    TDRV005_HANDLE          hdl
    UCHAR                   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

hdl

This value specifies the device handle 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;
```

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
{
    ULONG       Value;
    ULONG       Status;
} TDRV005_VALUE_BUF;
```

Value

This parameter holds the preload channel value.

Status

This parameter is not used for this function.

EXAMPLE

```
#include "tdrv005api.h"

TDRV005_HANDLE          hdl;
TDRV005_STATUS          result;
TDRV005_MULTIPLEVALUES  Values;

/*
** preload channel 0 and channel 5
*/
Values[0].Value = 0x00000000;
Values[5].Value = 0x50000000;
result = tdrv005globalCounterPreloadSet( hdl,
                                         TDRV005_CH0 | TDRV005_CH5,
                                         Values );
if (result != TDRV005_OK)
{
    /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified, data pointer is NULL.
TDRV005_ERR_NOT_RUNNING	A specified channel is not configured to counter mode.

3.6.4 tdrv005globalCounterLoad

NAME

tdrv005globalCounterLoad() – globally loads preload registers into counters.

SYNOPSIS

```
TDRV005_STATUS tdrv005globalCounterLoad
(
    TDRV005_HANDLE      hdl
    UCHAR               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

hdl

This value specifies the device handle 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.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;

/*
** load channel 0 and channel 5
*/
result = tdrv005globalCounterLoad(  hdl,
                                    TDRV005_CH0 | TDRV005_CH5 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified, data pointer is NULL.
TDRV005_ERR_NOT_RUNNING	A specified channel is not configured to counter mode.

3.6.5 tdrv005globalMultipleChannelRead

NAME

tdrv005globalMultipleChannelRead() – reads multiple channels simultaneously.

SYNOPSIS

```
TDRV005_STATUS tdrv005globalMultipleChannelRead
(
    TDRV005_HANDLE          hdl
    UCHAR                   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

hdl

This value specifies the device handle 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;
```

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
{
      ULONG         Value;
      ULONG         Status;
} TDRV005_VALUE_BUF;
```

Value

This parameter holds the returned channel value.

Status

This parameter holds the returned channel status.

EXAMPLE

```
#include "tdrv005api.h"

TDRV005_HANDLE          hdl;
int                     result;
TDRV005_MULTIPLEVALUES  MultipleValues;

/*
** read channel 0 and channel 5 simultaneously
*/
result = tdrv005globalMultipleChannelRead( hdl,
                                           TDRV005_CH0 | TDRV005_CH5,
                                           &MultipleValues );
if (result == TDRV005_OK)
{
     printf( "Channel(0) = 0x%08lX\n", MultipleValues.Channel[0].Value );
     printf( "Channel(5) = 0x%08lX\n", MultipleValues.Channel[5].Value );
}
else
{
     /* handle error */
}
```

RETURNS

TDRV005_OK if the function has been executed successfully, otherwise an error code.

ERROR CODES

Error code	Description
TDRV005_ERR_INVALID_HANDLE	Invalid device handle specified.
TDRV005_ERR_CHRNG	Invalid channel specified.
TDRV005_ERR_INVAL	Invalid parameter specified, data pointer is NULL.
TDRV005_ERR_BUSY	A MultipleChannelRead job is already pending, or a channel is busy with an SSI job.
TDRV005_ERR_TIMEOUT	Internal timeout. The values couldn't be retrieved within 2 seconds.