The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV005-SW-82

Linux Device Driver

6 Channel SSI, Incremental Encoder, Counter

Version 2.1.x

User Manual

Issue 2.1.0

October 2020

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-82

6 Channel SSI, Incremental Encoder, Counter

Linux Device Driver

Supported Modules:
TPMC117
TPMC317

Issue	Description	Date
1.0.0	First Issue	January 5, 2006
1.0.1	New address TEWS LLC, file list extended	June 18, 2008
2.0.0	API interface modified	December 16, 2013
2.0.1	File-list modified, General revision	February 18, 2019
2.1.0	Support additional features of TPMC117 (V2.0) added Functions: tdrv005GetBoardInfo(), tdrv005DebounceSetup() modified Functions: tdrv005TimerSetup() – (Argument: ClockFrequency)	October 12, 2020

Table of Contents

1 Introduction

The TDRV005-SW-82 Linux device driver software allows the operation of the TPMC117 family PMC conforming to the Linux 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 TDRV005-SW-82 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-82 supports the modules listed below:

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

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

2 Installation

Following files are located on the distribution media:

Directory path 'TDRV005-SW-82':

TDRV005-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
TDRV005-SW-82-2.1.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

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

The GZIP compressed archive TDRV005-SW-82-SRC.tar.gz contains the following files and directories:

Directory path 'tdrv005':

tdrv005.c	TDRV005 device driver source
tdrv005def.h	TDRV005 driver include file
tdrv005.h	TDRV005 include file for driver and application
Makefile	Device driver make file
makenode	Script for device node creation
COPYING	Copy of the GNU Public License (GPL)
api/tdrv005api.h	API include file
api/tdrv005api.c	API source file
example/tdrv005exa.c	Example application
example/Makefile	Example application makefile
include/config.h	Driver independent library header file
include/tpmodule.h	Driver and kernel independent library header file
include/tpmodule.c	Driver and kernel independent library source file
include/tpxxxhwdep.h	HAL library header file
include/tpxxxhwdep.c	HAL library source file

In order to perform an installation, extract all files of the archive TDRV005-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TDRV005-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tdrv005.h and tdrv005api.h to */lib/modules/<version>/build/include* and */usr/include*

2.1 Build and Install the Device Driver

- Login as *root*
- Change to the target directory
- To create and install the driver in the module directory */lib/modules/<version>/misc* enter:

  # make install

- To update the device driver's module dependencies, enter:

  # **depmod –aq**

2.2 Uninstall the Device Driver

- Login as *root*
- Change to the target directory
- To remove the driver from the module directory */lib/modules/<version>/misc* enter:

  # make uninstall

2.3 Install Device Driver into the Running Kernel

- To load the device driver into the running kernel, login as root and execute the following commands:

  # modprobe tdrv005drv

- After the first build or if you are using dynamic major device allocation it's necessary to create new device nodes on the file system. Please execute the script file *makenode,* which resides in the target directory, to do this. If your kernel has enabled a device file system (devfs or sysfs with udev) then skip running the *makenode* script. Instead of creating device nodes from the script the driver itself takes creating and destroying of device nodes in its responsibility.

  # sh makenode

On success the device driver will create a minor device for each compatible channel found. The first module can be accessed with device node /dev/tdrv005_0, the second PMC with device node /dev/tdrv005_1 and so on. The assignment of device nodes to physical PMC modules depends on the search order of the PCI bus driver.

2.4 Remove Device Driver from the Running Kernel

- To remove the device driver from the running kernel login as root and execute the following command:

  # modprobe –r tdrv005drv

If your kernel has enabled a device file system system (devfs or sysfs with udev) all /dev/tdrv005_x nodes will be automatically removed from your file system after this.

Be sure that the driver isn't opened by any application program. If opened you will get the response "*tdrv005drv: Device or resource busy*" and the driver will still remain in the system until you close all opened files and execute *modprobe –r* again.

2.5 Change Major Device Number

This paragraph is only for Linux kernels without DEVFS installed.

The TDRV005 device driver uses dynamic allocation of major device numbers per default. If this isn't suitable for the application it is possible to define a major number for the driver.

To change the major number edit the file tdrv005def.h, change the following symbol to appropriate value and enter `make install` to create a new driver.

TDRV005_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

EXAMPLE:

```
#define TDRV005_MAJOR   122
```

Be sure that the desired major number isn't used by other drivers. Please check *./proc/devices* to see which numbers are free.

Keep in mind that it is necessary to create new device nodes if the major number for the TDRV005 driver has changed and the makenode script isn't used.

3 API Documentation

3.1 General Functions

3.1.1 tdrv005Open

NAME

tdrv005Open – opens a device.

SYNOPSIS

```
TDRV005_HANDLE tdrv005Open
(
    char                    *DeviceName
)
```

DESCRIPTION

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

The tdrv005Open function can be called multiple times (e.g. in different tasks)

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TDRV005 device is named “/dev/tdrv005_0” the second device is named “/dev/tdrv005_1” and so on.

EXAMPLE

```
#include <tdrv005api.h>

TDRV005_HANDLE hdl;

/*
** open the specified device
*/
hdl = tdrv005Open(“/dev/tdrv005_0”);

if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

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

ERROR CODES

The error codes are stored in *errno.*

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

3.1.2 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 the device
*/
result = tdrv005Close(hdl);

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

RETURNS

TDRV005_OK on success or an appropriate system error code.

ERROR CODES

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

3.1.3 tdrv005GetPciInfo

NAME

tdrv005GetPciInfo – get information of the module PCI header

SYNOPSIS

```
TDRV005_STATUS tdrv005GetPciInfo
(
    TDRV005_HANDLE          hdl,
    TDRV005_PCIINFO_BUF     *pPciInfoBuf
)
```

DESCRIPTION

This function returns information of the module PCI header in the provided data buffer.

PARAMETERS

hdl

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

pPciInfoBuf

This argument is a pointer to the structure TDRV005_PCIINFO_BUF that receives information of the module PCI header.

```
typedef struct
{
    unsigned short      vendorId;
    unsigned short      deviceId;
    unsigned short      subSystemId;
    unsigned short      subSystemVendorId;
    int                 pciBusNo;
    int                 pciDevNo;
    int                 pciFuncNo;
} TDRV005_PCIINFO_BUF;
```

vendorId

PCI module vendor ID.

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module sub system vendor ID

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE          hdl;
TDRV005_STATUS          result;
TDRV005_PCIINFO_BUF     pciInfoBuf

/*
** get module PCI information
*/
result = tdrv005GetPciInfo( hdl, &pciInfoBuf );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.4 tdrv005GetBoardInfo

NAME

tdrv005GetBoardInfo – get information from the module

SYNOPSIS

```
TDRV005_STATUS tdrv005GetBoardInfo
(
    TDRV005_HANDLE      hdl,
    unsigned int        *firmwareId,
    int                 *temperature
)
```

DESCRIPTION

This function returns information about the module, e.g. firmware id (version) and the core temperature.

PARAMETERS

hdl

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

firmwareId

This argument points to an unsigned 32-bit buffer, where firmware id (version) will be copied to.

temperature

This argument points to a 32-bit buffer, where the current core temperature of the FPGA will be stored to. The temperature is returned in steps of 1/256 centigrade.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE          hdl;
TDRV005_STATUS          result;
unsigned int            firmId;
int                     healthTemp;

/*
** get module board information
*/
result = tdrv005GetBoardInfo( hdl, &firmId, &healthTemp );
if (result != TDRV005_OK)
{
     /* handle error */
}

printf(“Firmware-ID: %02d.%02d.%02d Build: %02d.\n”,
     (firmId >> 24) & 0xFF,
     (firmId >> 16) & 0xFF,
     (firmId >> 8) & 0xFF,
     firmId & 0xFF);
printf(“Temperature: %4d°C\n”, healthTemp);
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV005_ERR_NOTSUP	Function not supported (check board version)

3.1.5 tdrv005DebounceSetup

NAME

tdrv005DebounceSetup – configure debounce behavior of digital or SSI / Counter input lines

SYNOPSIS

```
TDRV005_STATUS tdrv005DebounceSetup
(
    TDRV005_HANDLE          hdl,
    TDRV005_DEBOUNCE_SEL    FunctionGroup,
    unsigned int            Channels,
    TDRV005_DEBOUNCE_TBASE  DebounceBase,
    unsigned int            DebounceCounter
)
```

DESCRIPTION

This function configures the 3-step debouncing behavior of all digital or all SSI / Counter input lines. The debounce time for the selected channels will be set. The debouncing behavior of unselected digital or SSI / Encoder channels will be set to default behavior (100ns).

PARAMETERS

hdl

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

FunctionGroup

This value specifies if the debounce behavior of the digital inputs or the SSI / Counter inputs shall be configured. Valid values are predefined:

Value	Description
TDRV005_DEBOUNCE_SEL_SSICOUNT	configure SSI / COUNTER debouncing
TDRV005_DEBOUNCE_SEL_DIGITAL	configure digital input debouncing

Channels

This value selects the channels which should be used with the debounce timer. The pre-defined values (TDRV005_CH0 – TDRV005_CH5) must be used. Multiple channels may be OR'ed to one value.

DebounceBase

This parameter specifies the timer base of the debouncer timer. The following values are defined and can be used:

Value	Description
TDRV005_DEBOUNCE_TBASE_31NS	time base: 31.25 ns
TDRV005_DEBOUNCE_TBASE_100NS	time base: 100 ns
TDRV005_DEBOUNCE_TBASE_1US	time base: 1 µs
TDRV005_DEBOUNCE_TBASE_1MS	time base: 1 ms

DebounceCounter

This value specifies the debounce counter value. The limits of the debounce time are calculated with the following formulas:

$T_{debounce(min)}$ = DebounceBase * DebounceCounter * 3

$T_{debounce(max)}$ = DebounceBase * DebounceCounter * 4

The value can be specified between 0 and 65535. If 0 is specified debouncing of the selected channels will be disabled.

EXAMPLE

```
#include “tdrv005api.h”

TDRV005_HANDLE          hdl;
TDRV005_STATUS          result;

/*
** define a debounce time for digital channel 0 and channel 3 of about 30µs
** channel 1, 2, 4 and channel 5 use default debouncing (100ns)
*/
result = tdrv005DebounceSetup (  hdl,
                                 TDRV005_DEBOUNCE_SEL_DIGITAL,
                                 TDRV005_CH0 | TDRV005_CH3,
                                 TDRV005_DEBOUNCE_TBASE_1US,
                                 10 );
if (result != TDRV005_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV005_ERR_INVAL	Invalid parameter specified
TDRV005_ERR_NOTSUP	Function does not support older hardware versions (TPMC117-xx V1.x and TPMC317-xx V1.x)

3.2 SSI Functions

3.2.1 tdrv005SsiSetup

NAME

tdrv005SsiSetup – sets up a channel for SSI operation.

SYNOPSIS

```
TDRV005_STATUS tdrv005SsiSetup
(
    TDRV005_HANDLE          hdl,
    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

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;
    unsigned char           NumberOfDataBits;
    TDRV005_SSI_CODING      Coding;
    unsigned char           ZeroBits;
    TDRV005_SSI_PARITY      Parity;
    unsigned char           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(hdl, TDRV005_CH0, &Options);

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

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
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,
    unsigned char       Channel,
    int                 Timeout,
    unsigned int        *Data,
    unsigned int        *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 unsigned int value where the data register content is stored.

Status

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

EXAMPLE

```
#include <tdrv005api.h>

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
unsigned int       ssiValue;
unsigned int       ssiStatus;

/*
** read the current counter value
*/
result = tdrv005SsiRead(hdl,
                        TDRV005_CH0,
                        TDRV005_WAIT_FOREVER,
                        &ssiValue,
                        &ssiStatus);

if (result == TDRV005_OK)
{
     printf( “SSI Value  = 0x%08X\n”, ssiValue );
     printf( “SSI Status = 0x%08X\n”, ssiStatus );
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	Channel is not configured to SSI mode, or there is another job in progress.

3.3 Counter Functions

3.3.1 tdrv005CounterSetup

NAME

tdrv005CounterSetup – sets up a channel for counter operation.

SYNOPSIS

```
TDRV005_STATUS tdrv005CounterSetup
(
    TDRV005_HANDLE          hdl,
    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

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
{
    unsigned char           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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
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,
    unsigned char       Channel,
    unsigned int        *Data,
    unsigned int        *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 unsigned int value where the data register content is stored.

Status

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

EXAMPLE

```
#include <tdrv005api.h>

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
unsigned int       CounterValue, Status;

/*
** read the current counter value
*/
result = tdrv005CounterRead( hdl,
                             TDRV005_CH0,
                             &CounterValue,
                             &Status);

if (result == TDRV005_OK)
{
     printf( “Counter Value = 0x%08X\n”, CounterValue );
     printf( “Status  Value = 0x%08X\n”, Status );
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
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,
    unsigned char       Channel,
    unsigned int        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;
unsigned int       PreloadValue;

/*
** load counter value
*/
PreloadValue = 0x12345678;

result = tdrv005CounterPreloadSet(hdl, TDRV005_CH0, PreloadValue);

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

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
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,
    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

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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
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,
    unsigned char       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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
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,
    unsigned char       Channel,
    unsigned int        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;
unsigned int       MatchValue;

/*
** wait indefinitely for counter match event
*/
MatchValue = 0x12345678;

result = tdrv005CounterWaitMatch(hdl,
                                 TDRV005_CH0,
                                 MatchValue,
                                 TDRV005_WAIT_FOREVER);

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

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	Channel is not configured to counter mode, or a counter-match job is already pending.
TDRV005_ERR_TIMEOUT	The event has not occurred, timeout.

3.3.7 tdrv005CounterWaitControlModeEvent

NAME

tdrv005CounterWaitControlModeEvent – waits for a counter-control-mode event.

SYNOPSIS

```
TDRV005_STATUS tdrv005CounterWaitControlModeEvent
(
    TDRV005_HANDLE      hdl,
    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

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;
TDRV005_STATUS     result;

/*
** wait indefinitely for counter control mode event
*/
result = tdrv005CounterWaitControlModeEvent(hdl,
                                            TDRV005_CH0,
                                            TDRV005_WAIT_FOREVER);

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

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	Channel is not configured to counter mode, or a counter-match job is already pending.
TDRV005_ERR_TIMEOUT	The event has not occurred, timeout.

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,
    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

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 or time base used as clock source for the timer. Possible values are:

Value	Description
TDRV005_CLKFRQ_1MHZ	clock frequency: 1 MHz clock
TDRV005_CLKFRQ_2MHZ	clock frequency: 2 MHz clock
TDRV005_CLKFRQ_4MHZ	clock frequency: 4 MHz clock
TDRV005_CLKFRQ_8MHZ	clock frequency: 8 MHz clock
TDRV005_CLKFRQ_BASE100NS	time base: 100 ns (only TPMC117 V2.x and newer)
TDRV005_CLKFRQ_BASE1US	time base: 1 µs (only TPMC117 V2.x and newer)
TDRV005_CLKFRQ_BASE1MS	time base: 1 ms (only TPMC117 V2.x and newer)
TDRV005_CLKFRQ_BASE1S	time base: 1 ns (only TPMC117 V2.x and newer)

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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_NOT_CONFIG	The timer was not 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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_NOT_CONFIG	The timer was not configured properly.

3.4.4 tdrv005TimerRead

NAME

tdrv005TimerRead – reads the current timer value.

SYNOPSIS

```
TDRV005_STATUS tdrv005TimerRead
(
      TDRV005_HANDLE                hdl,
      unsigned short                *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 unsigned short value where the data register content is stored.

EXAMPLE

```
#include <tdrv005api.h>

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
unsigned short     TimerValue;

/*
** read the current interval timer value
*/
result = tdrv005TimerRead(hdl, &TimerValue);

if (result == TDRV005_OK)
{
     printf( “Timer Value = 0x%04X\n”, TimerValue );
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.

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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_NOT_CONFIG	The timer was not configured properly.
TDRV005_ERR_TIMEOUT	The timer event has not occurred, timeout.
TDRV005_ERR_NOT_RUNNING	The timer is not running.

3.4.6 tdrv005TimerMultipleChannelReadSetup

NAME

tdrv005TimerMultipleChannelReadSetupt – sets up channels for multiple read.

SYNOPSIS

```
TDRV005_STATUS tdrv005TimerMultipleChannelReadSetup
(
    TDRV005_HANDLE      hdl,
    unsigned char       Channels
)
```

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.

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.

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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	Specified channel is busy with an SSI job.

3.4.7 tdrv005TimerMultipleChannelReadWait

NAME

tdrv005TimerMultipleChannelReadWait – waits for the simultaneous data timer event.

SYNOPSIS

```
TDRV005_STATUS tdrv005TimerMultipleChannelReadWait
(
    TDRV005_HANDLE          hdl,
    TDRV005_MULTIPLEVALUES  *MultipleValues,
    unsigned int            *Timestamp,
    int                     *MoreDataAvailable
)
```

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 timeout (1 second) 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
{
    unsigned int    Value;
    unsigned int    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 unsigned int value where the number of occurred timer interrupts is returned.

MoreDataAvailable

This is a pointer to a boolean integer value. The value is TRUE if additional data is available. This might happen if the timer event triggering the multiple-channel-read operation appears too fast. An additional call to tdrv005timerMultipleChannelReadWait must be performed.

EXAMPLE

```
#include <tdrv005api.h>

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

/*
** read channels triggered by timer
*/
result = tdrv005TimerMultipleChannelReadWait(hdl,
                                             &MultipleValues,
                                             &Timestamp,
                                             &MoreDataAvailable);

if (result == TDRV005_OK)
{
     printf( “Timestamp  = %d\n”, Timestamp);
     printf( “Channel(0) = 0x%08X\n”, MultipleValues.Channel[0].Value);
     printf( “Channel(5) = 0x%08X\n”, MultipleValues.Channel[5].Value);
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	Another job is already pending.
TDRV005_ERR_TIMEOUT	The timer event has not occurred, timeout.
TDRV005_ERR_NOT_CONFIG	The timer was not configured properly.

3.5 Digital Input Functions

3.5.1 tdrv005DigitalRead

NAME

tdrv005DigitalRead – reads the digital input lines.

SYNOPSIS

```
TDRV005_STATUS tdrv005DigitalRead
(
    TDRV005_HANDLE      hdl,
    unsigned char       *Data
)
```

DESCRIPTION

Read 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 *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.

EXAMPLE

```
#include <tdrv005api.h>

TDRV005_HANDLE     hdl;
TDRV005_STATUS     result;
unsigned char      DigitalValue;

/*
** read the current interval timer value
*/
result = tdrv005DigitalRead(hdl, &DigitalValue);

if (result == TDRV005_OK)
{
     printf( “Digital Value = 0x%02X\n”, DigitalValue );
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.

3.5.2 tdrv005DigitalWait

NAME

tdrv005DigitalWait – waits for an event on a digital input line.

SYNOPSIS

```
TDRV005_STATUS tdrv005DigitalWait
(
    TDRV005_HANDLE          hdl,
    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

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 transition on which the specified event should occur.

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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_TIMEOUT	The 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,
    unsigned char       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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.

3.6.2 tdrv005GlobalChannelDisable

NAME

tdrv005GlobalChannelDisable – globally disables multiple channels.

SYNOPSIS

```
TDRV005_STATUS tdrv005GlobalChannelDisable
(
    TDRV005_HANDLE      hdl,
    unsigned char       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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.

3.6.3 tdrv005GlobalCounterPreloadSet

NAME

tdrv005GlobalCounterPreloadSet – globally sets counter preload registers.

SYNOPSIS

```
TDRV005_STATUS tdrv005GlobalCounterPreloadSet
(
    TDRV005_HANDLE          hdl,
    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

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
{
        unsigned int        Value;
        unsigned int        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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	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,
    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

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 */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
TDRV005_ERR_BUSY	Channel is not configured to counter mode.

3.6.5 tdrv005GlobalMultipleChannelRead

NAME

tdrv005GlobalMultipleChannelRead – reads multiple channels simultaneously.

SYNOPSIS

```
TDRV005_STATUS tdrv005GlobalMultipleChannelRead
(
    TDRV005_HANDLE              hdl,
    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

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
{
    unsigned int    Value;
    unsigned int    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;
TDRV005_STATUS          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%08X\n", MultipleValues.Channel[0].Value);
    printf("Channel(5) = 0x%08X\n", MultipleValues.Channel[5].Value);
}
else
{
    /* handle error */
}
```

RETURN VALUE

On success, TDRV005_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERRORS

Error Code	Description
TDRV005_ERR_INVALID_HANDLE	The specified device handle is invalid.
TDRV005_ERR_INVAL	Invalid parameter specified.
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.

4 Diagnostic

If the TDRV005 device does not work properly, it is helpful to get some status information from the driver respective kernel.

The Linux */proc* file system provides information about kernel, resources, driver, devices, and so on. The following screen dumps displays information of a correct running TDRV005 driver (see also the proc man pages).

```
# lspci -v
…
04:02.0 Signal processing controller: TEWS Technologies GmbH Device 0075
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: medium devsel, IRQ 17
        Memory at feb9fc00 (32-bit, non-prefetchable) [size=128]
        I/O ports at e880 [size=128]
        Memory at feb9f800 (32-bit, non-prefetchable) [size=256]
        Kernel driver in use: TEWS TECHNOLOGIES - TDRV005 Device Driver
…

# cat /proc/devices
Character devices:
  1 mem
…
251 tdrv005drv
…


# cat /proc/iomem
…
  feb00000-febfffff : PCI Bus 0000:04
    feb9f800-feb9f8ff : 0000:04:02.0
      feb9f800-feb9f8ff : TDRV005
…
```