The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV016-SW-65

Windows Device Driver

32 / 16 Channels of 16 bit D/A

Version 2.0.x

User Manual

Issue 2.0.0

June 2013

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

TDRV016-SW-65

Windows Device Driver

32 / 16 Channels of 16 bit D/A

Supported Modules:

TPMC553
TPMC554

Issue	Description	Date
1.0.0	First Issue	December 13, 2010
1.0.1	General Revision	July 19, 2012
2.0.0	Device Handle Parameter, Return Values and Error Codes changed	June 21, 2013

Table of Contents

1 Introduction

The TDRV016-SW-65 Windows device driver is a kernel mode driver which allows the operation of the supported hardware module 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 standard file and device (I/O) functions (CreateFile, CloseHandle and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

The TDRV016-SW-65 device driver supports the following features:

- Configuration of DAC channel voltage ranges
- Configuration of Q-DAC operation modes (I/M/T- or F-Mode)
- Write analog output values in I- and M-Mode
- Use analog sequencer modes (T- or F-Mode)

The TDRV016-SW-65 device driver supports the modules listed below:

TPMC553	32 / 16 Channels of 16 bit D/A	(PMC)
TPMC554	32 / 16 Channels of 16 bit D/A with memory	(PMC)

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

TPMC553/554 User Manual
TPMC553/554 Engineering Manual

2 Installation

Following files are located in directory TDRV016-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)
tdrv016.inf	Windows installation script
tdrv016.h	Header file with IOCTL codes and structure definitions
api\tdrv016api.h	API include file
api\tdrv016api.c	API source file
example\tdrv016exa.c	Example application
TDRV016-SW-65-2.0.0.pdf	This document
Release.txt	Information about the Device Driver Release
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV016-SW-65 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 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 TDRV016 product family.

7. Copy needed files (tdrv016.h, API files) to desired target directory.

After successful installation a device is created for each module found (TDRV016_1, TDRV016_2 ...).

2.1.2 Windows 7

This section describes how to install the TDRV016-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 (tdrv016.h, API files) to desired target directory.

After successful installation a device is created for each module found (TDRV016_1, TDRV016_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 - TDRV016 (32 / 16 Channel 16-Bit DAC) (TPMC55x)***" should appear for each installed device.

3 API Documentation

3.1 General Functions

3.1.1 tdrv016Open

NAME

tdrv016Open() – opens a device.

SYNOPSIS

```
TDRV016_HANDLE tdrv016Open
(
    char  *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle 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. The first TDRV016 device is named "\\\\.\\TDRV016_1" the second device is named "\\\\.\\TDRV016_2" and so on.

EXAMPLE

```
#include "tdrv016api.h"
TDRV016_HANDLE hdl;

/*
** open the specified device
*/
hdl = tdrv016Open("\\\\.\\TDRV016_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 tdrv016Close

NAME

tdrv016Close() – closes a device.

SYNOPSIS

```
TDRV016_STATUS tdrv016Close
(
    TDRV016_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 “tdrv016api.h”
TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** close the device
*/
result = tdrv016Close( hdl );
if (result != TDRV016_OK)
{
     /* handle close error */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 tdrv016GetModuleInfo

NAME

tdrv016GetModuleInfo – get module information

SYNOPSIS

```
TDRV016_STATUS tdrv016GetModuleInfo
(
        TDRV016_HANDLE      hdl,
        int                 *pModuleType,
        int                 *pModuleVariant,
        int                 *pPciBusNo,
        int                 *pPciDevNo
)
```

DESCRIPTION

This function reads the currently configured voltage range of a specific D/A channel.

PARAMETERS

hdl

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

pModuleType

This argument is a pointer to an *int* value where the Module Type is returned. Possible values are:

Value	**Description**
TDRV016_MODTYPE_TPMC553	TPMC553
TDRV016_MODTYPE_TPMC554	TPMC554

pModuleVariant

This argument is a pointer to an *int* value where the Module Variant is returned. Possible values are:

Value	**Description**
10	-10 (32 D/A channels)
11	-11 (16 D/A channels)

pPciBusNo

This argument is a pointer to an *int* value where the PCI Bus number of the module is returned.

pPciDevNo

This argument is a pointer to an *int* value where the PCI Device number of the module is returned.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;
int                ModuleType;
int                ModuleVariant;
int                PciBusNo;
int                PciDevNo;

/*
** Read module information
*/
result = tdrv016GetModuleInfo(   hdl,
                                 &ModuleType,
                                 &ModuleVariant,
                                 &PciBusNo,
                                 &PciDevNo);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
     printf(“Module Type   : %d\n”, ModuleType);
     printf(“Module Variant: %d\n”, ModuleVariant);
     printf(“Localization  : Bus %d / Device %d\n”, PciBusNo, PciDevNo);
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid

3.2 Device Access Functions

3.2.1 tdrv016SetVoltageRange

NAME

tdrv016SetVoltageRange – set voltage range

SYNOPSIS

```
TDRV016_STATUS tdrv016SetVoltageRange
(
        TDRV016_HANDLE      hdl,
        int                 DacChannel,
        int                 VoltageRange,
        int                 Polarity
)
```

DESCRIPTION

This function configures the voltage range of a specific D/A channel.

PARAMETERS

hdl

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

DacChannel

This argument specifies the DAC channel number. Possible values are 1 up to the available number of channels for the specific module.

VoltageRange

This argument specifies the desired voltage range for the selected DAC channel. Possible values are:

Value	Description
TDRV016_VOLTRANGE_5V	Vmax = 5V
TDRV016_VOLTRANGE_10V	Vmax = 10V
TDRV016_VOLTRANGE_10P8V	Vmax = 10.8V

Polarity

This argument specifies the desired polarity for the selected DAC channel. Possible values are:

Value	Description
TDRV016_POLARITY_UNIPOL	0V .. +Vmax
TDRV016_POLARITY_BIPOL	-Vmax .. +Vmax

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** Configure DAC channel 3 to -10V .. +10V
*/
result = tdrv016SetVoltageRange( hdl,
                                 3,
                                 TDRV016_VOLTRANGE_10V,
                                 TDRV016_POLARITY_BIPOL);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid channel or parameter specified
TDRV016_ERR_IO	Error during hardware configuration

3.2.2 tdrv016GetVoltageRange

NAME

tdrv016GetVoltageRange – get current voltage range

SYNOPSIS

```
TDRV016_STATUS tdrv016GetVoltageRange
(
      TDRV016_HANDLE      hdl,
      int                 DacChannel,
      int                 *pVoltageRange,
      int                 *pPolarity
)
```

DESCRIPTION

This function reads the currently configured voltage range of a specific D/A channel.

PARAMETERS

hdl

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

DacChannel

This argument specifies the DAC channel number. Possible values are 1 up to the available number of channels for the specific module.

pVoltageRange

This argument is a pointer to an *int* value where the configured voltage range for the selected DAC channel is returned. Possible values are:

Value	Description
TDRV016_VOLTRANGE_5V	Vmax = 5V
TDRV016_VOLTRANGE_10V	Vmax = 10V
TDRV016_VOLTRANGE_10P8V	Vmax = 10.8V

pPolarity

This argument is a pointer to an *int* value where the configured polarity for the selected DAC channel is returned. Possible values are:

Value	Description
TDRV016_POLARITY_UNIPOL	0V .. +Vmax
TDRV016_POLARITY_BIPOL	-Vmax .. +Vmax

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;
int                VoltageRange, Polarity

/*
** Read current voltage range configuration of DAC channel 3
*/
result = tdrv016GetVoltageRange( hdl,
                                 3,
                                 &VoltageRange,
                                 &Polarity);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid channel specified

3.2.3 tdrv016QDacConfig

NAME

tdrv016QDacConfig – configure Q-DAC mode

SYNOPSIS

```
TDRV016_STATUS tdrv016QDacConfig
(
      TDRV016_HANDLE      hdl,
      int                 QDacNumber,
      int                 QDacMode,
      int                 GlobalLoadMode
)
```

DESCRIPTION

This function configures the operation mode of a specific Q-DAC, which serves 4 single D/A channels.

PARAMETERS

hdl

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

QDacNumber

This argument specifies the Q-DAC number. Possible values are 1 up to the available number of Q-DACs for the specific module. Q-DAC 1 serves D/A channels 1 to 4, Q-DAC 2 serves D/A channels 5 to 8 and so on.

QDacMode

This argument specifies the desired operation mode for the selected Q-DAC. Possible values are:

Value	**Description**
TDRV016_QDACMODE_INSTANT	Instant Mode. DAC values are written immediately.
TDRV016_QDACMODE_MANUAL	Manual mode. DAC values are written after manual load operation.
TDRV016_QDACMODE_TIMER	Timer mode. DAC values are written in sequencer mode.

GlobalLoadMode

This argument specifies if the Q-DAC should synchronize to other Q-DACs. If TRUE, all selected Q-DACs are updated simultaneously. This parameter is only relevant for Manual mode.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** Configure Q-DAC 1 (D/A channel 1 to 4)
** - use Manual Mode without global synchronization
*/
result = tdrv016QDacConfig(  hdl,
                             1,
                             TDRV016_QDACMODE_MANUAL,
                             FALSE);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid channel or parameter specified
TDRV016_ERR_IO	Error during hardware configuration

3.2.4 tdrv016DacWrite

NAME

tdrv016DacWrite – Write one DAC value to a specific DAC channel

SYNOPSIS

```
TDRV016_STATUS tdrv016DacWrite
(
        TDRV016_HANDLE      hdl,
        int                 DacChannel,
        int                 DacValue,
        int                 Flags
)
```

DESCRIPTION

This function writes one DAC value to a specific DAC channel. This function is supported for channels configured to Instant or Manual Mode.

PARAMETERS

hdl

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

DacChannel

This argument specifies the DAC channel which shall be updated. Possible values are 1 up to the number of available DAC channels of the specific module.

DacValue

This argument specifies the new DAC value for the specified channel.

Flags

This argument specifies additional options for this DAC update. Possible OR'ed flags are:

Value	Description
TDRV016_CORR	Use data correction for this conversion.
TDRV016_LOAD	Perform Load Operation for corresponding Q-DAC (only if Q-DAC is in Manual Mode).

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE      hdl;
TDRV016_STATUS      result;

/*
** Write new DAC value to channel 1, use data correction.
*/
result = tdrv016DacWrite(    hdl,
                             1,
                             0x1000,
                             TDRV016_CORR);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid channel specified
TDRV016_ERR_ACCESS	Specified channel not configured to I- or M-Mode

3.2.5 tdrv016DacWriteMulti

NAME

tdrv016DacWriteMulti – Write DAC values to multiple DAC channels

SYNOPSIS

```
TDRV016_STATUS tdrv016DacWriteMulti
(
        TDRV016_HANDLE      hdl,
        unsigned int        DacChannelMask,
        unsigned int        CorrectionMask,
        int                 PerformLoad,
        unsigned short      DacData[32]
)
```

DESCRIPTION

This function writes different DAC value to specified DAC channels. This function is supported for channels configured to Instant or Manual Mode.

PARAMETERS

hdl

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

DacChannelMask

This argument specifies DAC channels which shall be updated. A set (1) bit specifies that the corresponding channel shall be updated. Bit 0 corresponds to the first DAC channel, bit 1 corresponds to the second DAC channel and so on.

CorrectionMask

This argument specifies if data correction shall be used for specific DAC channels. A set (1) bit enables data correction for the corresponding channel. Bit 0 corresponds to the first DAC channel, bit 1 corresponds to the second DAC channel and so on.

PerformLoad

This argument specifies if the corresponding Q-DACs shall be updated. If TRUE, all affected Q-DACs are updated using the Load Operation. If this parameter is FALSE, all Q-DACs configured to Manual Mode will not be updated.

DacData

This argument specifies the new DAC data. Array index 0 corresponds to the first DAC channel, array index 1 corresponds to the second DAC channel and so on. Only channels marked for update using parameter *DacChannelMask* will be modified.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;
unsigned short     DacData[32];

/*
** Write new DAC values to channel 1, 2 and 32.
** Use data correction only for channel 1.
** Update all channels which are in M-Mode.
*/
DacData[0]     = 0x1000;
DacData[1]     = 0x2000;
DacData[31]    = 0x0000;

result = tdrv016DacWriteMulti(    hdl,
                             ((1 << 31) | (1 << 1) | (1 << 0)),
                             (1 << 0),
                             TRUE,
                             DacData);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_ACCESS	At least one of the specified channels is not in I- or M-Mode
TDRV016_ERR_IO	Error during hardware configuration

3.2.6 tdrv016QDacLoad

NAME

tdrv016QDacLoad – Perform Load Operation for specified Q-DACs

SYNOPSIS

```
TDRV016_STATUS tdrv016QDacLoad
(
        TDRV016_HANDLE      hdl,
        unsigned int        QDacMask
)
```

DESCRIPTION

This function performs the Load Operation for specified Q-DACs, to achieve simultaneous update of multiple DAC channels. This function is supported for Q-DACs configured to Manual Mode.

PARAMETERS

hdl

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

QDacMask

This argument specifies the Q-DACs which shall be loaded. A set (1) bit specifies that the corresponding Q-DAC shall be loaded. Bit 0 corresponds to the first Q-DAC (DAC channels 1 to 4), bit 1 corresponds to the second Q-DAC (DAC channels 5 to 8) and so on.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** Load Q-DACs 1 and 8 simultaneously.
*/
result = tdrv016QDacLoad(    hdl,
                             ((1 << 7) | (1 << 0)));
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_ACCESS	At least one of the Q-DACs is not in M-Mode.

3.3 Sequencer Functions

3.3.1 tdrv016SequencerConfig

NAME

tdrv016SequencerConfig – configure sequencer cycle time

SYNOPSIS

```
TDRV016_STATUS tdrv016SequencerConfig
(
        TDRV016_HANDLE      hdl,
        int                 QDacNumber,
        unsigned int        CycleTime
)
```

DESCRIPTION

This function configures the sequencer cycle time of a specific Q-DAC, which serves 4 single D/A channels. The configured sequencer cycle time is used in both Timer and FIFO mode.

PARAMETERS

hdl

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

QDacNumber

This argument specifies the Q-DAC number. Possible values are 1 up to the available number of Q-DACs for the specific module. Q-DAC 1 serves D/A channels 1 to 4, Q-DAC 2 serves D/A channels 5 to 8 and so on.

CycleTime

This argument specifies the sequencer cycle time. The sequencer timer is configurable in steps of 10µs. Possible values are 0 to the maximum value specified in the corresponding module hardware user manual.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** Configure Sequencer Timer of Q-DAC 1 (D/A channel 1 to 4)
** Use 1ms cycle time.
*/
result = tdrv016SequencerConfig( hdl,
                            1,
                            99);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid Q-DAC specified.

3.3.2 tdrv016SequencerStart

NAME

tdrv016SequencerStart – start sequencer timer

SYNOPSIS

```
TDRV016_STATUS tdrv016SequencerStart
(
        TDRV016_HANDLE      hdl,
        unsigned int        SequencerMask
)
```

DESCRIPTION

This function starts the sequencer timer of specified Q-DACs. This function starts the sequencer operation of both Timer and FIFO mode.

PARAMETERS

hdl

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

SequencerMask

This argument specifies the Q-DACs which shall be started in sequencer mode. A set (1) bit specifies that the corresponding Q-DAC shall be started. Bit 0 corresponds to the first Q-DAC (DAC channels 1 to 4), bit 1 corresponds to the second Q-DAC (DAC channels 5 to 8) and so on.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** Start Sequencer Timer of Q-DAC 1 and 2
*/
result = tdrv016SequencerStart( hdl, (1 << 1) | (1 << 0) );

if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_ACCESS	At least one of the Q-DACs is not in Timer or FIFO mode.

3.3.3 tdrv016SequencerStop

NAME

tdrv016SequencerStop – stop sequencer timer

SYNOPSIS

```
TDRV016_STATUS tdrv016SequencerStop
(
        TDRV016_HANDLE      hdl,
        unsigned int        SequencerMask
)
```

DESCRIPTION

This function stops the sequencer timer of specified Q-DACs. This function stops the sequencer operation of both Timer and FIFO mode.

PARAMETERS

hdl

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

SequencerMask

This argument specifies the Q-DACs which shall be stopped. A set (1) bit specifies that the corresponding Q-DAC shall be stopped. Bit 0 corresponds to the first Q-DAC (DAC channels 1 to 4), bit 1 corresponds to the second Q-DAC (DAC channels 5 to 8) and so on.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;

/*
** Stop Sequencer Timer of Q-DAC 2
*/
result = tdrv016SequencerStop( hdl, (1 << 1) );

if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid

3.3.4 tdrv016SequencerWrite

NAME

tdrv016SequencerWrite – Write DAC data in Timer mode

SYNOPSIS

```
TDRV016_STATUS tdrv016SequencerWrite
(
    TDRV016_HANDLE      hdl,
    int                 QDacNumber,
    unsigned int        UseCorrection,
    int                 DacValue[4],
    int                 timeout
)
```

DESCRIPTION

This function writes new DAC data to a specific Q-DAC, which serves 4 single D/A channels. All four channels of the Q-DAC are affected. This function is only supported in Timer Mode. This function might block until the next sequencer interrupt allows transferrin the DAC data, or the timeout expires.

PARAMETERS

hdl

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

QDacNumber

This argument specifies the Q-DAC number. Possible values are 1 up to the available number of Q-DACs for the specific module. Q-DAC 1 serves D/A channels 1 to 4, Q-DAC 2 serves D/A channels 5 to 8 and so on.

UseCorrection

This argument specifies if data correction shall be used for specific DAC channels. A set (1) bit enables data correction for the corresponding channel. Bit 0 corresponds to the first DAC channel of the Q-DAC, bit 1 corresponds to the second DAC channel of the Q-DAC and so on.

DacValue

This argument specifies the new DAC data. Array index 0 corresponds to the first DAC channel of the Q-DAC, array index 1 corresponds to the second DAC channel of the Q-DAC and so on.

timeout

This parameter specifies the time the function will block until the data is transferred into the DAC channels, which is done using interrupts. The interrupts are raised based upon the configured sequencer cycle time, so this timeout value must be chosen according to the configured cycle time. This timeout value is specified in milliseconds. The resulting time depends on the system tick granularity. To wait indefinitely, specify -1.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;
int                DacData[4];

/*
** Write new data to Q-DAC 2 without data correction.
** Use 500ms for timeout.
*/
result = tdrv016SequencerWrite(  hdl,
                                 2,
                                 0,
                                 DacData,
                                 500);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid Q-DAC specified.
TDRV016_ERR_ACCESS	Specified Q-DAC is not in Timer-Mode.
TDRV016_ERR_TIMEOUT	Timeout while waiting for sequencer interrupt.

3.4 FIFO Functions

3.4.1 tdrv016FifoConfig

NAME

tdrv016FifoConfig – configure FIFO mode (TPMC554 only)

SYNOPSIS

```
TDRV016_STATUS tdrv016FifoConfig
(
      TDRV016_HANDLE      hdl,
      unsigned int        DacChannelMask,
      unsigned int        ContinuousModeMask,
      int                 Size[32],
      int                 Limit[32]
)
```

DESCRIPTION

This function configures the FIFO mode of specified DAC channels. All four channels of one affected Q-DAC are used in FIFO mode. All channels which were previously configured to FIFO mode and are not again configured with this function are configured to Instant mode without changing the DAC value.

PARAMETERS

hdl

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

DacChannelMask

This argument specifies the DAC channels which shall be used in FIFO mode. All four DAC channels of one affected Q-DAC must be configured for FIFO mode. A set (1) bit specifies that the corresponding channel shall be configured. Bit 0 corresponds to the first DAC channel, bit 1 corresponds to the second DAC channel and so on.

ContinuousModeMask

This argument specifies if the corresponding DAC channel FIFO shall be used in continuous mode. A set (1) configures the corresponding channel to repeat its FIFO data. An unset (0) bit configures the channel to stop the data output if the FIFO runs empty. Bit 0 corresponds to the first DAC channel, bit 1 corresponds to the second DAC channel and so on.

Size

This argument specifies the size of the FIFO in number of values. Array index 0 corresponds to the first DAC channel, array index 1 corresponds to the second DAC channel and so on.

Limit

This argument specifies the FIFO trigger limit where an interrupt is raised. The limit is specified as 2^{Limit}. Array index 0 corresponds to the first DAC channel, array index 1 corresponds to the second DAC channel and so on.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;
int                Size[32];
int                Limit[32];


/*
** Configure FIFO mode for channel 1 to 8 (Q-DAC 1 and 2)
** - Use DAC 1 and 4 in Continuous Mode
*/
Size[0]   = 100;
Limit[0]  = 5;     /* Limit at 32 values */
...

result = tdrv016FifoConfig(  hdl,
                             0x000000ff,
                             (1 << 3) | (1 << 0),
                             Size,
                             Limit);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Not all DAC channels of a Q-DAC specified, invalid Size or Limit.
TDRV016_ERR_ACCESS	Module does not support FIFO mode.
TDRV016_ERR_IO	Error during hardware configuration

3.4.2 tdrv016FifoWrite

NAME

tdrv016FifoWrite – Write DAC data in FIFO mode (TPMC554 only)

SYNOPSIS

```
TDRV016_STATUS tdrv016FifoWrite
(
    TDRV016_HANDLE      hdl,
    int                 DacChannel,
    unsigned int        Flags,
    int                 NumValues,
    unsigned short      *pDacData,
    int                 timeout
)
```

DESCRIPTION

This function writes new DAC data of a specific DAC channel into the FIFO. This function is only supported in FIFO Mode. The function blocks until all data is written into the FIFO, or the timeout expires.

PARAMETERS

hdl

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

DacChannel

This argument specifies the DAC channel number. Possible values are 1 up to the available number of DACs for the specific module.

Flags

This argument specifies additional options for this DAC update. Possible value:

Value	**Description**
TDRV016_CORR	Use data correction for all values.

NumValues

This argument specifies the number of DAC data values which shall be written into the FIFO.

pDacData

This parameter points to the DAC data section where the specified number of 16bit values is stored.

timeout

This parameter specifies the time the function will block until the data is transferred into the FIFO, which might be done using interrupts. The interrupts are raised based upon the configured sequencer cycle time, so this timeout value must be chosen according to the configured cycle time. This timeout value is specified in milliseconds. The resulting time depends on the system tick granularity. To wait indefinitely, specify -1.

The timeout value specifies the time to wait for the next interrupt. Multiple interrupts might be required to transfer the complete amount of specified data into the FIFO.

EXAMPLE

```
#include “tdrv016api.h”

TDRV016_HANDLE     hdl;
TDRV016_STATUS     result;
unsigned short     DacData[100];

/*
** Write 100 DAC data values to FIFO 2 with data correction.
** Use 500ms for timeout.
*/
DacData[0] = 0x1234;
DacData[1] = 0x5678;
DacData[2] = 0x9ABC;
...

result = tdrv016FifoWrite(   hdl,
                             2,
                             TDRV016_CORR,
                             100,
                             DacData,
                             500);
if (result != TDRV016_OK)
{
     /* handle error */
}
else
{
     /* successful */
}
```

RETURNS

On success, TDRV016_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	**Description**
TDRV016_ERR_INVALID_HANDLE	The specified device handle is invalid
TDRV016_ERR_INVAL	Invalid Channel specified.
TDRV016_ERR_ACCESS	Module does not support FIFO mode, or channel is not in FIFO mode.
TDRV016_ERR_BUSY	This DAC channel is already busy transferring data into the FIFO.
TDRV016_ERR_TIMEOUT	Timeout while waiting for FIFO interrupt.