The Embedded I/O Company

TEWS TECHNOLOGIES

TIP102-SW-42

VxWorks Device Driver

Motion Controller with Incremental Encoder Interface

Version 2.0.x

User Manual

Issue 2.0.0

November 2008

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

TEWS TECHNOLOGIES LLC
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com
Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TIP102-SW-42

VxWorks Device Driver

Motion Controller with Incremental Encoder Interface

Supported Modules:
TIP102

Issue	Description	Date
1.0	First Issue (Driver implemented as function interface)	August 3, 2000
1.1	Support for x86 with SBS PCI40 Carrier added	February 8, 2002
2.0.0	New driver implementation with TEWS Carrier Support, Support of VxWorks I/O system, Application Interface implemented	November 20, 2008

Table of Contents

1 INTRODUCTION 4
1.1 Device Driver 4
1.2 IPAC Carrier Driver 5
2 INSTALLATION 6
2.1 Include device driver in Tornado IDE project 6
2.2 Special installation for Intel x86 based targets 6
2.3 System resource requirement 7
3 I/O SYSTEM FUNCTIONS 8
3.1 tip102Drv() 8
3.2 tip102DevCreate() 10
4 I/O FUNCTIONS 13
4.1 open() 13
4.2 close() 15
4.3 ioctl() 17
4.3.1 FIO_TIP102_READ_COUNTER 19
4.3.2 FIO_TIP102_PRESET_COUNTER 20
4.3.3 FIO_TIP102_CONFIGURE 21
4.3.4 FIO_TIP102_WAIT_FOR_INT 24
4.3.5 FIO_TIP102_READ_ADC 25
4.3.6 FIO_TIP102_WRITE_DAC 26
4.3.7 FIO_TIP102_READ_REGISTER 28
4.3.8 FIO_TIP102_WRITE_REGISTER 29
5 API DOCUMENTATION 32
5.1 General Functions 32
5.1.1 tip102open() 32
5.1.2 tip102close() 33
5.2 Encoder Functions 35
5.2.1 tip102ReadCounter() 35
5.2.2 tip102SetCounterPreset() 36
5.2.3 tip102ConfigureCounter() 38
5.2.4 tip102waitForTrigger() 41
5.3 Analog I/O Functions 43
5.3.1 tip102ReadADC() 43
5.3.2 tip102SetDAC() 44
5.4 Direct Register Access Functions 46
5.4.1 tip102ReadRegister() 46
5.4.2 tip102SetRegister 48

1 Introduction

1.1 Device Driver

The TIP102-SW-42 VxWorks device driver software allows the operation of the supported IPAC conforming to the VxWorks I/O system specification. This includes a device-independent basic I/O interface with *open*(), *close(),*and *ioctl()* functions.

The TIP102-SW-42 device driver supports the following features:

- Configure encoder counter
- Read counter value
- Set preset counter value
- Wait for counter event
- Read ADC input value
- Write DAC output value
- Read and Write registers directly

The TIP102-SW-42 supports the modules listed below:

TIP102-1x	1 Channel Motion Controller (Incremental)	IndustryPack® compatible
TIP102-2x	2 Channel Motion Controller (Incremental)	IndustryPack® compatible

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

TIP102 User manual

TIP102 Engineering Manual

CARRIER-SW-42 IPAC Carrier User Manual

1.2 IPAC Carrier Driver

IndustryPack (IPAC) carrier boards have different implementations of the system to IndustryPack bus bridge logic, different implementations of interrupt and error handling and so on. Also the different byte ordering (big-endian versus little-endian) of CPU boards will cause problems on accessing the IndustryPack I/O and memory spaces.

To simplify the implementation of IPAC device drivers which work with any supported carrier board, TEWS TECHNOLOGIES has designed a so called Carrier Driver that hides all differences of different carrier boards under a well defined interface.

The TEWS TECHNOLOGIES IPAC Carrier Driver CARRIER-SW-42 is part of this TIP102-SW-42 distribution. It is located in directory CARRIER-SW-42 on the corresponding distribution media.

This IPAC Device Driver requires a properly installed IPAC Carrier Driver. Due to the design of the Carrier Driver, it is sufficient to install the IPAC Carrier Driver once, even if multiple IPAC Device Drivers are used.

Please refer to the CARRIER-SW-42 User Manual for a detailed description how to install and setup the CARRIER-SW-42 device driver, and for a description of the TEWS TECHNOLOGIES IPAC Carrier Driver concept.

How to use the carrier driver in the application program is shown in the programming example tip102exa.c.

If the IPAC carrier driver isn't used for the IPAC driver setup, the application software has to setup carrier board hardware, mapping of device memory and interrupt level setup by itself.

2 Installation

Following files are located on the distribution media:

Directory path 'TIP102-SW-42':

tip102drv.c	TIP102 device driver source
tip102def.h	TIP102 driver include file
tip102.h	TIP102 include file for driver and application
tip102api.c	TIP102 application interface (API)
tip102api.h	TIP102 API include file for application
tip102exa.c	Example application
include/tdhal.h	Hardware dependent interface functions and definitions
ipac_carrier.h	Carrier driver interface definitions
TIP102-SW-42-2.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

2.1 Include device driver in Tornado IDE project

For including the TIP102-SW-42 device driver into a Tornado IDE project follow the steps below:

(1) Copy the files from the distribution media into a subdirectory in your project path.
(For example: ./TIP102)

(2) Add the device drivers C-files to your project.
Make a right click to your project in the 'Workspace' window and use the 'Add Files ...' topic.
A file select box appears, and the driver files can be selected.

(3) Now the driver is included in the project and will be built with the project.

For a more detailed description of the project facility please refer to your Tornado User's Guide.

2.2 Special installation for Intel x86 based targets

The TIP102 device driver is fully adapted for Intel x86 based targets. This is done by conditional compilation directives inside the source code and controlled by the VxWorks global defined macro **CPU_FAMILY**. If the content of this macro is equal to *I80X86* special Intel x86 conforming code and function calls will be included.

2.3 System resource requirement

The table gives an overview over the system resources that will be needed by the driver.

Resource	Driver requirement	Devices requirement
Memory	< 1 KB	< 1 KB
Stack	< 1 KB	---
Semaphores	---	<number of channels>

Memory and Stack usage may differ from system to system, depending on the used compiler and its setup.

The following formula shows the way to calculate the common requirements of the driver and devices.

*<total requirement> = <driver requirement> + (<number of devices> * <device requirement>)*

The maximum usage of some resources is limited by adjustable parameters. If the application and driver exceed these limits, increase the according values in your project.

3 I/O system functions

This chapter describes the driver-level interface to the I/O system. The purpose of these functions is to install the driver in the I/O system, add and initialize devices.

3.1 tip102Drv()

NAME

tip102Drv() - installs the TIP102 driver in the I/O system

SYNOPSIS

#include “tip102.h”

STATUS tip102Drv(void)

DESCRIPTION

This function initializes the TIP102 driver and installs it in the I/O system.

A call to this function is the first thing the user has to do before adding any device to the system or performing any I/O request.

EXAMPLE

```
#include "tip102.h”

STATUS            result;

/*-------------------
  Initialize Driver
  -------------------*/
result = tip102Drv();
if (result == ERROR)
{
     /* Error handling */
}
```

RETURNS

OK or ERROR. If the function fails an error code will be stored in *errno*.

ERROR CODES

Error codes are only set by system functions. The error codes are stored in *errno* and can be read with the function *errnoGet()*.

SEE ALSO

VxWorks Programmer's Guide: I/O System

3.2 tip102DevCreate()

NAME

tip102DevCreate() – Add a TIP102 device to the VxWorks system

SYNOPSIS

```
#include “tip102.h”

STATUS tip102DevCreate
(
        char        *name,
        int         devIdx,
        int         funcType,
        void        *pParam
)
```

DESCRIPTION

This routine adds the selected device to the VxWorks system. The device hardware will be setup and prepared for use.

This function must be called before performing any I/O request to this device.

PARAMETER

name

This string specifies the name of the device that will be used to identify the device, for example for *open()* calls.

devIdx

This index number specifies the TIP102 minor device number to add to the system.

If modules of the same type are installed the device numbers will be assigned in the order the IPAC CARRIER *ipFindDevice()* function will find the devices.

For TIP102 devices there is only one devIdx per hardware module starting with devIdx = 0.

funcType

This parameter is unused and should be set to *0*.

pParam

This parameter points to a structure (*TIP102_DEVCONFIG*) containing the default configuration of the device.

The structure (*TIP102_DEVCONFIG*) has the following layout and is defined in tip102.h:

```
typedef struct
{
    struct ipac_resource *ipac;
} TIP102_DEVCONFIG;
```

ipac

Pointer to TIP102 module resource descriptor, retrieved by CARRIER Driver ipFindDevice() function

EXAMPLE

```
#include "tip102.h"

STATUS                  result;
TIP102_DEVCONFIG        tip102Conf;
struct ipac_resource    ipac;

/* IPAC CARRIER Driver initialization */

/*
**   Find an IP module from TEWS TECHNOLOGIES (manufacturer = 0xB3)
**   with model number MODEL_TIP102_2x (see tip102.h).
**   This module uses both interrupt lines and needs an IACK cycle,
**   we need only the IO space base address for the related driver.
**   !!!! If TIP102-1x and TIP102-2x are used both, the application
**   !!!! will have to search for both model numbers
*/
result = ipFindDevice(  0xB3, MODEL_TIP102_2x, 0,
                        IPAC_INT0_EN | IPAC_INT1_EN | IPAC_LEVEL_SENS |
                        IPAC_IACK_CYC | IPAC_CLK_8MHZ, &ipac);

if (result == OK)
{
     devConfig.ipac = &ipac;

     …
```

```
    …

    /*-------------------------------------------------------
      Create the device "/tip102/0" for the first device
      -------------------------------------------------------*/
    tip102Conf.ipac =  &ipac;

    result = tip102DevCreate(    "/tip102/0",
                                 0,
                                 0,
                                 (void*)&tip102Conf);
    if (result == OK)
    {
         /* Device successfully created */
    }
    else
    {
         /* Error occurred when creating the device */
    }
}
else
{
    /* No IP found on supported IP carrier boards */
}
```

RETURNS

OK or ERROR. If the function fails an error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno* and can be read with the function *errnoGet()*.

Error code	Description
S_ioLib_NO_DRIVER	The driver has not been started
EINVAL	Invalid input argument
EISCONN	The device has already been created
ENOTSUP	IPAC with unsupported model number specified
ETIMEDOUT	Initial A/D Conversion timed out

SEE ALSO

VxWorks Programmer's Guide: I/O System

4 I/O Functions

4.1 open()

NAME

open() - open a device or file.

SYNOPSIS

```
int open
(
        const char  *name,
        int         flags,
        int         mode
)
```

DESCRIPTION

Before I/O can be performed to the TIP102 device, a file descriptor must be opened by invoking the basic I/O function *open().*

PARAMETER

name

Specifies the device which shall be opened, the name specified in tip102DevCreate() must be used

flags

Not used

mode

Not used

EXAMPLE

```
int      fd;

/*----------------------------------------
  Open the device named "/tip102/0" for I/O
  ----------------------------------------*/
fd = open("/tip102/0", 0, 0);
if (fd == ERROR)
{
     /* Handle error */
}
```

RETURNS

A device descriptor number or ERROR. If the function fails an error code will be stored in *errno*.

ERROR CODES

The error code can be read with the function *errnoGet()*.

The error code is a standard error code set by the I/O system (see VxWorks Reference Manual).

SEE ALSO

ioLib, basic I/O routine - *open()*

4.2 close()

NAME

close() – close a device or file

SYNOPSIS

```
STATUS close
(
     int          fd
)
```

DESCRIPTION

This function closes opened devices.

PARAMETER

fd

This file descriptor specifies the device to be closed. The file descriptor has been returned by the *open()* function.

EXAMPLE

```
int           fd;
STATUS        retval;

/*----------------
  close the device
  ----------------*/
retval = close(fd);
if (retval == ERROR)
{
     /* Handle error */
}
```

RETURNS

OK or ERROR. If the function fails, an error code will be stored in *errno*.

ERROR CODES

The error code can be read with the function *errnoGet()*.

The error code is a standard error code set by the I/O system (see VxWorks Reference Manual).

SEE ALSO

ioLib, basic I/O routine - close()

4.3 ioctl()

NAME

ioctl() - performs an I/O control function.

SYNOPSIS

#include “tip102.h”

```
int ioctl
(
        int     fd,
        int     request,
        int     arg
)
```

DESCRIPTION

Special I/O operation that do not fit to the standard basic I/O calls (read, write) will be performed by calling the ioctl() function.

PARAMETER

fd

This file descriptor specifies the device to be used. The file descriptor has been returned by the *open()* function.

request

This argument specifies the function that shall be executed. Following functions are defined:

Function	Description
FIO_TIP102_READ_COUNTER	Read current encoder counter value
FIO_TIP102_PRESET_COUNTER	Set new preset value
FIO_TIP102_CONFIGURE	Configure counter
FIO_TIP102_WAIT_FOR_INT	Wait for configured counter event
FIO_TIP102_READ_ADC	Read ADC value
FIO_TIP102_WRITE_DAC	Write DAC value
FIO_TIP102_READ_REGISTER	Read the value of a specified register
FIO_TIP102_WRITE_REGISTER	Write a value to a specified register

arg

This parameter depends on the selected function (request). How to use this parameter is described below with the function.

RETURNS

Function dependent value (described with the function) or ERROR. If the function fails an error code will be stored in *errno*.

ERROR CODES

The error code can be read with the function *errnoGet()*.

The error code is a standard error code set by the I/O system (see VxWorks Reference Manual).

SEE ALSO

ioLib, basic I/O routine - ioctl()

4.3.1 FIO_TIP102_READ_COUNTER

This I/O control function reads the current counter value of the specified encoder channel. The function specific control parameter **arg** is a pointer on a TIP102_COUNTER_BUF structure.

```
typedef struct
{
        int             channelNo;
        long            value;
} TIP102_COUNTER_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel

value

This field returns the counter value. (Value range: -8388608 … 8388607)

EXAMPLE

```
#include “tip102.h”

int                 fd;
TIP102_COUNTER_BUF cntBuf;
int                 retval;

…

/*------------------------------
  Get counter value of channel 1
  ------------------------------*/
cntBuf.channelNo = 1;

retval = ioctl(fd, FIO_TIP102_READ_COUNTER, (int)&cntBuf);
if (retval != ERROR)
{
     /* function succeeded */
     printf("Counter: %ld\n", cntBuf.value);
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number)

4.3.2 FIO_TIP102_PRESET_COUNTER

This I/O control function sets the new counter preset value of the specified encoder channel. The function specific control parameter **arg** is a pointer on a TIP102_COUNTER_BUF structure.

```
typedef struct
{
    int         channelNo;
    long        value;
} TIP102_COUNTER_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel

value

This value specifies the new counter preset value. (Value range: -8388608 … 8388607)

EXAMPLE

```
#include “tip102.h”

int                fd;
TIP102_COUNTER_BUF cntBuf;
int                retval;

/*-----------------------------------------
  Set new Preset value of channel 1 to -10000
  -----------------------------------------*/
cntBuf.channelNo = 1;
cntBuf.value = -10000;

…
```

…

```
retval = ioctl(fd, FIO_TIP102_PRESET_COUNTER, (int)&cntBuf);
if (retval != ERROR)
{
     /* function succeeded */
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number or preset value)

4.3.3 FIO_TIP102_CONFIGURE

This I/O control function configures the counter of the specified encoder channel. The function specific control parameter **arg** is a pointer on a TIP102_CONFIG_BUF structure.

For detailed configuration information, please refer to the TIP102 User Manual.

```
typedef struct
{
     int            channelNo;
     unsigned char  refMode;
     unsigned char  quadMode;
     unsigned char  encInpDiff;
     unsigned char  cyMode;
     unsigned char  bcdMode;
} TIP102_CONFIG_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel

refMode

This value specifies the reference mode. One of the following modes must be selected for this parameter:

Reference Mode	Description
TIP102_REF_NONE	disable reference logic
TIP102_REF_NOSWITCH	the zero signal will trigger the load counter input
TIP102_REF_SWITCH	the zero signal is combined with the reference switch input
TIP102_REF_EXTTRIG	the external input trigger will be used

The flag *TIP102_AUTO_REF* can be ORed binary to the selected mode if the auto reference shall be enabled.

quadMode

This value specifies the quadrature mode. One of the following modes must be specified:

Quadrature Mode	Description
TIP102_XOFF	counter is disabled
TIP102_X1	counter has single resolution of the encoder
TIP102_X2	counter has double resolution of the encoder
TIP102_X4	counter has quadruple resolution of the encoder

encInpDiff

This value specifies if the input signal is TTL or differential. For TTL the value must be FALSE (0), for differential it must be TRUE (not 0).

cyMode

This value specifies the configuration of the Trigger Input mode. The following modes are defined:

Trigger Input Mode	Description
TIP102_CY_NOTCY	The Trigger input is the CY# signal (carry)
TIP102_CY_CYT	The Trigger input is the CYT signal (carry toggle flip flop)
TIP102_CY_CY	The Trigger input is the CY signal (carry)
TIP102_CY_NOTCOMP	The Trigger input is the COMP# signal (compare)

bcdMode

This value specifies if a binary or BCD counter is used. For a binary counter the value must be FALSE (0), for a BCD counter it must be TRUE (not 0).

EXAMPLE

```
#include “tip102.h”

int                fd;
TIP102_CONFIG_BUF  confBuf;
int                retval;

…

/*----------------------------------------
  Configure channel 1 (see settings below)
  ----------------------------------------*/
confBuf.channelNo =     1;
confBuf.refMode =       TIP102_REF_SWITCH | /* with reference switch    */
                        TIP102_AUTO_REF;    /* enable auto reference    */
confBuf.quadMode =      TIP102_X2;          /* double resolution        */
confBuf.encInpDiff =    FALSE;              /* TTL - no differential    */
confBuf.cyMode =        TIP102_CY_NOTCOMP;  /* compare is trigger input */
confBuf.bcdMode =       FALSE;              /* binary mode              */

retval = ioctl(fd, FIO_TIP102_CONFIG, (int)&confBuf);
if (retval != ERROR)
{
     /* function succeeded */
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (e.g. invalid channel number)

4.3.4 FIO_TIP102_WAIT_FOR_INT

This I/O control function waits for an interrupt event of the specified encoder channel. The event is configured with the Trigger Input Mode by the FIO_TIP102_CONFIGURE function. If the interrupt does not occur in a specified time, the function will return with an error. The function specific control parameter **arg** is a pointer on a TIP102_WAIT_BUF structure.

```
typedef struct
{
    int         channelNo;
    int         timeout;
} TIP102_WAIT_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

timeout

This value specifies the maximum time the function shall wait, before it times out. The time is specified in system ticks. If a value “WAIT_FOREVER” is specified the call will not time out.

EXAMPLE

```
#include “tip102.h”

int             fd;
TIP102_WAIT_BUF waitBuf;
int             retval;

/*-------------------------------------------------------------
  Wait for interrupt event on channel 1, timeout = 5000 ticks
  -------------------------------------------------------------*/
waitBuf.channelNo  = 1;
waitBuf.timeout    = 5000;

retval = ioctl(fd, FIO_TIP102_WAIT_FOR_INT, (int)&waitBuf);
if (retval != ERROR)
{
    /* function succeeded */
}
else
{
    /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number)

4.3.5 FIO_TIP102_READ_ADC

This I/O control function executes an A/D conversion on the specified channel and returns the converted value. The function specific control parameter **arg** is a pointer on a TIP102_ADC_DAC_BUF structure.

```
typedef struct
{
        int             channelNo;
        short           value;
} TIP102_ADC_DAC_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

value

This value returns the result of the A/D conversion. The value range will be between -2048 and 2047 (-10V...+10V).

EXAMPLE

```
#include “tip102.h”

int                 fd;
TIP102_ADC_DAC_BUF  adcBuf;
int                 retval;

/*----------------------------
  Get ADC value from channel 1
  ----------------------------*/
adcBuf.channelNo = 1;

…
```

…

```
retval = ioctl(fd, FIO_TIP102_READ_ADC, (int)&adcBuf);
if (retval != ERROR)
{
     /* function succeeded */
     printf("ADC-value: %d\n", adcBuf.value);
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number)
ETIMEDOUT	A/D conversion timed out (hardware defect possible)

4.3.6 FIO_TIP102_WRITE_DAC

This I/O control function starts a D/A conversion for the specified channel. The function specific control parameter **arg** is a pointer on a TIP102_ADC_DAC_BUF structure.

```
typedef struct
{
     int          channelNo;
     short        value;
} TIP102_ADC_DAC_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

value

This value specifies the (16-bit) DAC output value to be used. The value range must be between -32768 and 32767 (-10V...+10V).

EXAMPLE

```
#include “tip102.h”

int                fd;
TIP102_ADC_DAC_BUF dacBuf;
int                retval;

/*-----------------------------------
  Set DAC output for channel 1 to ~5V
  -----------------------------------*/
dacBuf.channelNo   = 1;
dacBuf.value       = 16384;  /* Fullscale/2 */

retval = ioctl(fd, FIO_TIP102_WRITE_DAC, (int)&dacBuf);
if (retval != ERROR)
{
     /* function succeeded */
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number)

4.3.7 FIO_TIP102_READ_REGISTER

This I/O control function reads the current of a specified register.

This function allows an application to use all functions and configurations of the encoder counter channel. Keep in mind that access to the TIP102 with this function may have influence on the behavior of other ioctl() functions.

The function specific control parameter **arg** is a pointer on a TIP102_REGISTER_BUF structure.

```
typedef struct
{
        int             channelNo;
        int             regOff;
        unsigned char   value;
} TIP102_REGISTER_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

regOff

This value specifies the register that shall be read. The following registers are accessible by this function:

Register Offset	Register
TIP102_REGOFF_INPSR	Input Status Register (INPSR)
TIP102_REGOFF_OUTCR	Output Control Register (OUTCR)
TIP102_REGOFF_CNTDA	Counter Data Register (CNTDA)
TIP102_REGOFF_CNTCS	Counter Control and Status Register (CNTCS)
TIP102_REGOFF__OSR	Output Status Register (CNTCS [OSR])

value

This field returns the current register value.

EXAMPLE

```
#include “tip102.h”

int                     fd;
TIP102_REGISTER_BUF     regBuf;
int                     retval;

…
```

```
…

/*-------------------------------------------------------------
  Get current content of the input status register of channel 1
  -------------------------------------------------------------*/
regBuf.channelNo   = 1;
regBuf.regOff      = TIP102_REGOFF_INPSR;

retval = ioctl(fd, FIO_TIP102_READ_REGISTER, (int)&regBuf);
if (retval != ERROR)
{
     /* function succeeded */
     printf("Register value: %02X\n", regBuf.value);
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number or register)

4.3.8 FIO_TIP102_WRITE_REGISTER

This I/O control function writes a new value into a specified register.

This function allows an application to use all functions and configurations of the encoder counter channel. Keep in mind that access to the TIP102 with this function may have influence on the behavior of other ioctl() functions.

The function specific control parameter **arg** is a pointer on a TIP102_REGISTER_BUF structure.

```
typedef struct
{
      int             channelNo;
      int             regOff;
      unsigned char   value;
} TIP102_REGISTER_BUF;
```

channelNo

Specifies the channel number (axis). Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel

regOff

This value specifies the register that shall be read. The following registers are accessible by this function:

Register Offset	Register
TIP102_REGOFF_INPSR	Input Status Register (INPSR)
TIP102_REGOFF_OUTCR	Output Control Register (OUTCR)
TIP102_REGOFF_CNTDA	Counter Data Register (CNTDA)
TIP102_REGOFF_CNTCS	Counter Control and Status Register (CNTCS)
TIP102_REGOFF__MCR	Master Control Register (CNTCS [MCR])
TIP102_REGOFF__ICR	Input Control Register (CNTCS [ICR])
TIP102_REGOFF__OCCR	Output Control Register (CNTCS [OCCR])
TIP102_REGOFF__QR	Quadrature Register (CNTCS [QR])

value

This parameter specifies the new register value.

EXAMPLE

```
#include “tip102.h”

int                     fd;
TIP102_REGISTER_BUF     regBuf;
int                     retval;

/*-----------------------------------------------
  Transfer preload value into counter channel 1
  -----------------------------------------------*/
regBuf.channelNo   = 1;
regBuf.regOff      = TIP102_REGOFF__MCR;
regBuf.value       = 0x08;

retval = ioctl(fd, FIO_TIP102_WRITE_REGISTER, (int)&regBuf);
if (retval != ERROR)
{
     /* function succeeded */
}
else
{
     /* handle the error */
}
```

ERROR CODES

Error code	Description
EINVAL	arg is NULL or invalid parameter value (invalid channel number or register)

5 API Documentation

5.1 General Functions

5.1.1 tip102open()

NAME

tip102open() – opens a device.

SYNOPSIS

```
int tip102open
(
      char          *DeviceName
)
```

DESCRIPTION

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

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device.

EXAMPLE

```
#include “tip102api.h”

int FileDescriptor;

/*
** open file descriptor to device
*/
FileDescriptor = tip102open(“/tip102_0”);
if (FileDescriptor == ERROR)
{
     /* handle open error */
}
```

RETURNS

A device descriptor number, or ERROR if the function fails an error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno* and can be read with the function *errnoGet()*.

The error code is a standard error code set by the I/O system (see VxWorks Reference Manual).

5.1.2 tip102close()

NAME

tip102close() – closes a device.

SYNOPSIS

```
int tip102close
(
      int     FileDescriptor
)
```

DESCRIPTION

This function closes previously opened devices.

PARAMETERS

FileDescriptor

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

EXAMPLE

```
#include “tip102api.h”

int FileDescriptor;
int result;

…
```

…

```
/*
** close file descriptor to device
*/
result = tip102close(FileDescriptor);
if (result == ERROR)
{
     /* handle close error */
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno* and can be read with the function *errnoGet()*.

The error code is a standard error code set by the I/O system (see VxWorks Reference Manual).

5.2 Encoder Functions

5.2.1 tip102ReadCounter()

NAME

tip102ReadCounter() – Read counter value from TIP102 channel

SYNOPSIS

```
int tip102ReadCounter
(
        int     FileDescriptor,
        int     chanNo,
        int     *value
)
```

DESCRIPTION

This function reads the current value of specified encoder counter.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

value

This pointer refers to a buffer where the counter value will be returned.
(Value range: -8388608 … 8388607)

EXAMPLE

```
#include “tip102api.h”

int  FileDescriptor;
int  result;
int  value;

…
```

…

```
/* Read counter value from channel 2 */
result = tip102ReadCounter(FileDescriptor, 2, &value);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
     printf("    Counter = %d (%06Xh)\n", value, value & 0xFFFFFF);
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (invalid channel number)

5.2.2 tip102SetCounterPreset()

NAME

tip102SetCounterPreset() – Set preset value of specified encoder counter

SYNOPSIS

```
int tip102SetCounterPreset
(
      int     FileDescriptor,
      int     channel,
      int     value
)
```

DESCRIPTION

This function sets the preset value of specified encoder counter.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

value

This value specifies the new preset value. (Value range: -8388608 … 8388607)

EXAMPLE

```
#include “tip102api.h”

int  FileDescriptor;
int  result;

…

/* Set preset value of channel 2 t0 125000 */
result = tip102SetCounterPreset(FileDescriptor, 2, 125000);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (invalid channel number or preset value)

5.2.3 tip102ConfigureCounter()

NAME

tip102ConfigureCounter() – Configure TIP102 counter channel

SYNOPSIS

```
int tip102ConfigureCounter
(
        int     FileDescriptor,
        int     channel,
        int     quadMode,
        int     referenceMode,
        int     enaAutoReference,
        int     inpTrigMode,
        int     diffInpSignals,
        int     enaBcd
)
```

DESCRIPTION

This function configures the specified encoder counter.

For detailed configuration information, please refer to the TIP102 User Manual.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

quadMode

This value specifies the quadrature mode resolution.

Value	Description
0	counter is disabled
1	counter has single resolution of the encoder
2	counter has double resolution of the encoder
4	counter has quadruple resolution of the encoder

referenceMode

This value specifies the reference mode. One of the following modes must be selected for this parameter:

Value	Description
TIP102_REF_NONE	disable reference logic
TIP102_REF_NOSWITCH	the zero signal will trigger the load counter input
TIP102_REF_SWITCH	the zero signal is combined with the reference switch input
TIP102_REF_EXTTRIG	the external input trigger will be used

enaAutoReference

This value specifies if the automatic reference mode shall be enabled (TRUE) or not (FALSE). Automatic reference mode is only selectable if referenceMode is TIP102_REF_SWITCH or TIP102_REF_EXTTRIG

inpTrigMode

This value specifies the Trigger Input mode. The following modes are defined:

Value	Description
TIP102_CY_NOTCY	The Trigger input is the CY# signal (carry)
TIP102_CY_CYT	The Trigger input is the CYT signal (carry toggle flip flop)
TIP102_CY_CY	The Trigger input is the CY signal (carry)
TIP102_CY_NOTCOMP	The Trigger input is the COMP# signal (compare)

diffInpSignals

This value specifies if the input signal is TTL or differential. For TTL the value must be FALSE , for differential it must be TRUE.

enaBcd

This value specifies if a binary or BCD counter is used. For a binary counter the value must be FALSE, for a BCD counter it must be TRUE.

EXAMPLE

```
#include “tip102api.h”

int  FileDescriptor;
int  result;

…
```

```
…

/* Configure channel 1:
          X4
          none reference mode
          single-ended input
          COMP# is trigger input
          binary data */
result = tip102ConfigureCounter( FileDescriptor,
                                 1,
                                 4,
                                 TIP102_REF_NONE,
                                 FALSE,
                                 TIP102_CY_NOTCOMP,
                                 FALSE,
                                 FALSE);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (e.g. invalid channel number)

5.2.4 tip102waitForTrigger()

NAME

tip102waitForTrigger() – Wait for configured counter event

SYNOPSIS

```
int tip102waitForTrigger
(
        int     FileDescriptor,
        int     channel,
        int     timeout
)
```

DESCRIPTION

This function waits until the configured counter event occurs or the specified time has passed. If the time has passed, the function returns with an error.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

timeout

This value specifies the maximum time the function shall wait for the counter event. This value is specified in milliseconds. A value of -1 specifies that the function shall never timeout.

EXAMPLE

```
#include “tip102api.h”

int  FileDescriptor;
int  result;

…
```

…

```
/* Wait for the counter event on channel 1 , but timeout after a minute */
result = tip102waitForTrigger(FileDescriptor, 1, 60000);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
}

/* Wait for the counter event on channel 2 , there shall be no timeout */
result = tip102waitForTrigger(FileDescriptor, 2, -1);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (e.g. invalid channel number)

5.3 Analog I/O Functions

5.3.1 tip102ReadADC()

NAME

tip102ReadADC() – Read ADC value from TIP102 channel

SYNOPSIS

```
int tip102ReadADC
(
        int     FileDescriptor,
        int     chanNo,
        int     *value
)
```

DESCRIPTION

This function executes an A/D conversion and returns the converted value.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

value

This pointer refers to a buffer where the ADC value will be returned.
(Value range: -2048 … 2047)

EXAMPLE

```
#include “tip102api.h”

int  FileDescriptor;
int  result;
int  value;

…
```

…

```
/* Get input voltage on channel 2 */
result = tip102ReadADC(FileDescriptor, 2, &value);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
     printf("    Input voltage = %6.3fV\n", (value * 10.0) / 0x800);
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (e.g. invalid channel number)

5.3.2 tip102SetDAC()

NAME

tip102SetDAC() – Write DAC value to TIP102 channel

SYNOPSIS

```
int tip102SetDAC
(
      int     FileDescriptor,
      int     chanNo,
      int     value
)
```

DESCRIPTION

This function starts a D/A conversion.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

value

This value specifies the new output value. (Value range: -32768 … 32767)

EXAMPLE

```
#include “tip102api.h”

int  FileDescriptor;
int  result;

…

/* Set output voltage on channel 2 to 0 Volt */
result = tip102SetDAC(FileDescriptor, 2, 0);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (e.g. invalid channel number)

5.4 Direct Register Access Functions

5.4.1 tip102ReadRegister()

NAME

tip102ReadRegister() – Read from register

SYNOPSIS

```
int tip102ReadRegister
(
        int             FileDescriptor,
        int             chanNo,
        int             rdRegister,
        unsigned char   *value
)
```

DESCRIPTION

This function allows a direct read access from the specified register.

This function allows the application to use all functions and configurations of the encoder counter channel. Keep in mind that access to the TIP102 with this function may have influence on the behavior of counter.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

rdRegister

This value specifies the register that shall be read.

Value	Register
TIP102_REGOFF_INPSR	Input Status Register (INPSR)
TIP102_REGOFF_OUTCR	Output Control Register (OUTCR)
TIP102_REGOFF_CNTDA	Counter Data Register (CNTDA)
TIP102_REGOFF_CNTCS	Counter Control and Status Register (CNTCS)
TIP102_REGOFF__OSR	Output Status Register (CNTCS [OSR])

value

This pointer refers to a buffer where the current register value will be returned.

EXAMPLE

```
#include “tip102api.h”

int           FileDescriptor;
int           result;
unsigned char ucVal;

…

/* Read register content of the input status register of channel 1 */
result = tip102ReadRegister( FileDescriptor,
                             1,
                             TIP102_REGOFF_INPSR,
                             &ucValue);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
     printf("    Value = %02Xh\n", ucValue);
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (e.g. invalid channel number)

5.4.2 tip102SetRegister

NAME

tip102SetRegister() – Write to register

SYNOPSIS

```
int tip102SetRegister
(
        int             FileDescriptor,
        int             chanNo,
        int             wrRegister,
        unsigned char   value
)
```

DESCRIPTION

This function allows a direct write access to the specified register.

This function allows the application to use all functions and configurations of the encoder counter channel. Keep in mind that access to the TIP102 with this function may have influence on the behavior of counter.

PARAMETERS

FileDescriptor

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

chanNo

This value specifies the channel number (axis) on the TIP102. Allowed channel numbers are 1 and 2 for TIP102-2x with two channels or only 1 for TIP102-1x with one channel.

wrRegister

This value specifies the register that shall be written.

Value	Register
TIP102_REGOFF_INPSR	Input Status Register (INPSR)
TIP102_REGOFF_OUTCR	Output Control Register (OUTCR)
TIP102_REGOFF_CNTDA	Counter Data Register (CNTDA)
TIP102_REGOFF_CNTCS	Counter Control and Status Register (CNTCS)
TIP102_REGOFF__MCR	Master Control Register (CNTCS [MCR])
TIP102_REGOFF__ICR	Input Control Register (CNTCS [ICR])
TIP102_REGOFF__OCCR	Output Control Register (CNTCS [OCCR])
TIP102_REGOFF__QR	Quadrature Register (CNTCS [QR])

value

This parameter specifies the new register value.

EXAMPLE

```
#include “tip102api.h”

int           FileDescriptor;
int           result;

…

/* Set output control register of channel 1 to 0 */
result = tip102ReadRegister( FileDescriptor, 1, TIP102_REGOFF_OUTCR, 0);
if (result == ERROR)
{
     printf("ERROR (%08Xh)\n", errnoGet());
}
else
{
     printf("OK\n");
}
```

RETURNS

OK, or ERROR if the function fails, an error code will be stored in *errno*.

ERROR CODES

Error code	Description
EINVAL	invalid parameter value (e.g. invalid channel number)