The Embedded I/O Company

TEWS TECHNOLOGIES

TIP845-SW-42

VxWorks Device Driver

48 Channel 14 bit A/D converter

Version 2.1.x

User Manual

Issue 2.1.0

May 2010

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

TIP845-SW-42

VxWorks Device Driver

48 Channel 14 bit A/D converter

Supported Modules:
TIP845

Issue	Description	Date
1.0.0	First Issue	October 5, 2005
2.0.0	tip845DevCreate() parameters modified, Address TEWS LLC changed, New file list	July 27, 2007
2.0.1	Carrier Driver description added	June 24, 2008
2.1.0	Address TEWS LLC removed, tip845DevCreate changed	May 6, 2010

Table of Contents

1 Introduction

1.1 Device Driver

The TIP845-SW-42 VxWorks device driver software allows the operation of the TIP845 IPAC conforming to the VxWorks I/O system specification. This includes a device-independent basic I/O interface with *open*(), *close(),* and *ioctl()* functions.

Special I/O operation that do not fit to the standard I/O calls will be performed by calling the ioctl() function with a specific function code and an optional function dependent argument.

The TIP845-SW-42 device driver supports the following features:

- Reading data from a specified channel
- Reading data sets from specified channels in sequencer mode
- Starting and configuring sequencer mode
- Stopping sequencer mode
- Data correction while reading data values using the factory stored correction data
- Using single-ended or differential input interface
- Using input gain of 1,2,4 and 8

The TIP845-SW-42 supports the modules listed below:

TIP845	48 Channel 14 bit A/D converter	IndustryPack® compatible

To get more information about the features and use of TIP845 devices it is recommended to read the manuals listed below.

TIP845 User manual

TIP845 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 TIP845-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-65 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 tip845exa.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 'TIP845-SW-42':

tip845drv.c	TIP845 device driver source
tip845def.h	TIP845 driver include file
tip845.h	TIP845 include file for driver and application
tip845exa.c	Example application
include/ipac_carrier.h	Carrier driver interface definitions
TIP845-SW-42-2.1.0.pdf	PDF copy of this manual
Release.txt	Release information
ChangeLog.txt	Release history

2.1 Include device driver in Tornado IDE project

For Including the TIP845-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: ./TIP845)

(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 TIP845 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	~ 20 Bytes	~ 472 Bytes
Stack	< 100 Bytes	---
Semaphores	0	1

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

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 tip845Drv()

NAME

tip845Drv() - installs the TIP845 driver in the I/O system

SYNOPSIS

#include “tip845.h”

STATUS tip845Drv(void)

DESCRIPTION

This function installs the TIP845 driver in the I/O system.

The call of 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 "tip845.h”

/*-------------------
  Initialize Driver
  -------------------*/
status = tip845Drv();
if (status == 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 tip845DevCreate()

NAME

tip845DevCreate() – Add a TIP845 device to the VxWorks system

SYNOPSIS

#include “tip845.h”

```
STATUS tip845DevCreate
(
    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 device to add to the system.

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

For TIP845 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 is not used and should be set to NULL.

EXAMPLE

```
#include "tip845.h”

STATUS            result;

result = tip675DevCreate(    "/tip845/0",
                             0,
                             0,
                             NULL);
if (result == OK)
{
     /* Device successfully created */
}
else
{
     /* Error occurred when creating the device */
}
```

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	No driver installed
EIO	Access to the hardware failed
EBUSY	The Device has already been created
EINVAL	Invalid parameter value specified

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 TIP845 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 tip845DevCreate() must be used

flags

Not used

mode

Not used

EXAMPLE

```
int  fd;

/*----------------------------------------
  Open the device named "/tip845/0" for I/O
  ----------------------------------------*/
fd = open("/tip845/0", 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

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

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

A device descriptor number 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

ioLib, basic I/O routine - close()

4.3 ioctl()

NAME

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

SYNOPSIS

#include “tip845.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_TIP845_READ	Read ADC data from a specified channel
FIO_TIP845_SEQREAD	Wait for sequencer data and read a set of ADC data
FIO_TIP845_SEQSTART	Configure and start sequencer
FIO_TIP845_SEQSTOP	Stop sequencer

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 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) or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
ENOSYS	requested function not supplied

SEE ALSO

ioLib, basic I/O routine - ioctl()

4.3.1 FIO_TIP845_READ

This I/O control function starts an A/D conversion for specified channel, waits for completion and read the ADC value. Parameters specify if a single-ended or differential interface shall be used, which gain shall be used, and if the returned data shall be corrected using factory set correction data. If the function is used more than once with the same parameters, the driver will not use settling time, this will decrease the execution time of the function. The function specific control parameter **arg** is a pointer on a *TIP845_ADC_BUFFER* structure.

```
typedef struct
{
        int             channel;
        int             gain;
        unsigned long   flags;
        short           value;
} TIP845_ADC_BUFFER;
```

channel

This parameter specifies the channel to use for the next A/D conversion. Allowed channel numbers are 1 … 48 for single-ended conversions, and 1 ... 24 for differential conversions.

gain

This parameter specifies the input that shall be used for the next A/D conversion. Allowed gain values are 1, 2, 4, and 8.

flags

This parameter is an ORed value of the following flags (defined in tip845.h):

TIP845_FLAG_ADC_CORR	If this flag is set, the input data will be corrected with the factory stored correction data. If the flag is not set the data will be returned not corrected.
TIP845_FLAG_ADC_DIFF	If this flag is set, the differential input interface will be used. (channel numbers 1 … 24) If the flag is not set the singe-ended input interface will be used. (channel numbers 1 ... 48)

value

This parameter is used to return the data value. The data will be returned as a 16 bit (short) value. The data will be stored in the lower 14 bit and will be sign extended to 16 bit.

EXAMPLE

```
#include “tip845.h”

int                fd;
TIP845_ADC_BUFFER  readBuf;
unsigned long      retval;

/*------------------------
  Read A/D value
    - channel 12
    - differential interface
    - data correction
    - gain = 4
  ------------------------*/
readBuf.channel = 12;
readBuf.gain = 4;
readBuf.flags = TIP845_FLAG_ADC_CORR | TIP845_FLAG_ADC_DIFF;

retval = ioctl(fd, FIO_TIP845_READ, (int)&readBuf);
if (retval != ERROR)
{
     printf(“input value: %d\n”, readBuf.value);
}
else
{
     /* handle the error */
}
```

RETURN VALUE

OK if function succeeds or ERROR.

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) or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
EBUSY	Device is busy, sequencer mode is active.
EINVAL	Invalid argument specified
ENOTSUP	Channel number is not supported (with the specified interface)
EIO	Hardware access failed

4.3.2 FIO_TIP845_SEQREAD

This I/O control function reads a data set in sequencer mode. The function specific control parameter **arg** is a pointer on a *TIP845_SEQ_READ_BUFFER* structure.

```
typedef struct
{
        short                   value[48];
        unsigned long           status;
} TIP_SEQ_READ_BUFFER;
```

value[]

This array will be filled with A/D data. Disabled channel will return 0. For access to the array the define *TIP845_SEQ_IDX()* shall be used.
- for single-ended interfaces use :
  TIP845_SEQ_IDX(0, <channel number>)
- for differential interfaces use :
  TIP845_SEQ_IDX(1, <channel number>)

status

This parameter will return the actual state of the sequencer. The returned value is an ORed value of the following flags (defined in tip845.h):

TIP845_STATUS_IRAM_ERR	This flag is set if an instruction RAM error has been occurred
TIP845_STATUS_TIMER_ERR	This flag is set if a timer error has been occurred
TIP845_STATUS_OVERFLOW	This flag is set if an overflow error has been occurred

EXAMPLE

```
#include “tip845.h”

int                     fd;
TIP845_SEQ_READ_BUFFER  seqBuf;
unsigned long           retval;

/*-------------------------------------
  Wait for the next sequencer data set
  -------------------------------------*/
retval = ioctl(fd, FIO_TIP845_SEQREAD, (int)&seqBuf);

…
```

…

```
if (retval != ERROR)
{
     printf("Channel 5 (single-ended): &d\n",
          seqBuf.data[TIP845_SEQ_IDX(0,5)]);
     printf("Channel 5 (differential): &d\n",
          seqBuf.data[TIP845_SEQ_IDX(1,5)]);
}
else
{
     /* handle the error */
}
```

RETURN VALUE

OK if function succeeds or ERROR.

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) or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
EACCES	No access, sequencer is not started
EBUSY	An other task is already waiting for data
ETIMEDOUT	The function has timed out
EIO	The sequencer has detected a problem, status flags will be set.

4.3.3 FIO_TIP845_SEQSTART

This I/O control function configures and starts the sequencer. The function specific control parameter **arg** is a pointer on a *TIP845_SEQ_START_BUFFER* structure.

```
typedef struct
{
    TIP845_SEQ_SETUP_CHAN       sglChanConf[48];
    TIP845_SEQ_SETUP_CHAN       diffChanConf[24];
    unsigned short              seqTimer;
    long                        seqReadTimeout;
} TIP845_SEQ_START_BUFFER;
```

sglChanConf[]

This array contains configuration parameters for single-ended channels. An index of 0 selects channel 1, an index of 1 select channel 2 and so on.

diffChanConf[]

This array contains configuration parameters for differential channels. An index of 0 selects channel 1, an index of 1 select channel 2 and so on.

seqTimer

This value specifies the sequencer cycle time. The value is specified in step of 100µs. A value of 0 will enable the sequencers continuous mode.

seqReadTimeout

This value specifies a timeout time for sequencer read access. The timeout time is returned in 1ms steps. If the value is set to -1, the function will wait without timeout.

```
typedef struct
{
    int             gain;
    unsigned long   flags;
} TIP845_SEQ_SETUP_CHAN;
```

gain

This argument specifies the input gain that shall be used for the selected channel. Allowed gain values are 1, 2, 4, and 8.

flags

This parameter is an ORed value of the following flags (defined in tip845.h):

TIP845_FLAG_ADC_CORR	If this flag is set, the input data will be corrected with the factory stored correction data. If the flag is not set the data will be returned not corrected.
TIP845_FLAG_ADC_ENABLE	If this flag is set, the selected channel will be enabled in sequencer mode. If the flag is not set the selected channel will be disabled in sequencer mode.

Channel numbers for single-ended and differential interface are not compatible. Please refer to the TIP845 User Manual to get more information

EXAMPLE

```
#include “tip845.h”

int                          fd;
TIP845_SEQ_START_BUFFER      seqStartBuf;
unsigned long                retval;
int                          i;

for (i = 0; i < 24; i++)
{
     seqStartBuf.diffChanConf[i].flags = 0;     /* channel disable */
}
for (i = 0; i < 48; i++)
{
     seqStartBuf.snglChanConf[i].flags = 0;     /* channel disable */
}

/*-------------------------------------
  Start sequencer
  Sequencer cycle time: 1 sec
  Read Timeout:         2 sec
    Enable channel 5 single-ended
      gain:       2
      correction: off
    and channel 5 differential
      gain:       1
      correction: on
  (These both channels do not conflict)
  -------------------------------------*/
seqStartBuf.seqTimer         = 10000;  /* 1 sec */
seqStartBuf.seqReadTimeout   = 2000;   /* 2 sec */

/* configure sngl channel 5 disable */
seqStartBuf.snglChanConf[4].gain      = 2;
seqStartBuf.snglChanConf[4].flags     = TIP845_FLAG_ADC_ENABLE;

…
```

…

```
/* configure differential channel 5 disable */
seqStartBuf.diffChanConf[4].gain      = 1;
seqStartBuf.diffChanConf[4].flags     = TIP845_FLAG_ADC_ENABLE |
                                        TIP845_FLAG_ADC_CORR;

retval = ioctl(fd, FIO_TIP845_SEQSTART, (int)&seqStartBuf);
if (retval != ERROR)
{
     /* function succeeded, sequencer started */
}
else
{
     /* handle the error */
}
```

RETURN VALUE

OK if function succeeds or ERROR.

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) or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
EBUSY	Device is busy, sequencer is active
EINVAL	Invalid parameter specified, or function detected a conflict between enabled singe-ended and differential channels

4.3.4 FIO_TIP845_SEQSTOP

This I/O control function stops the sequencer. The function specific control parameter **arg** is not used for this function.

EXAMPLE

```
#include “tip845.h”

int               fd;
unsigned long     retval;

/*-------------------
  Stop sequencer mode
  -------------------*/
retval = ioctl(fd, FIO_TIP845_SEQSTO, 0);
if (retval != ERROR)
{
     /* function succeeded */
}
else
{
     /* handle the error */
}
```

RETURN VALUE

OK if function succeeds or ERROR.

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) or a driver set code described below. Function specific error codes will be described with the function.

Error code	Description
EACCES	Sequencer mode already disabled