The Embedded I/O Company

TEWS TECHNOLOGIES

TIP840-SW-82

Linux Device Driver

16 / 8 Channel 12 Bit ADC

Version 1.0.x

User Manual

Issue 1.0.0

March 2009

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

TIP840-SW-82

Linux Device Driver

16 / 8 Channel 12 Bit ADC

Supported Modules:
TIP840

Issue	Description	Date
1.0.0	First Issue	March 2, 2009

Table of Contents

1 Introduction

The TIP840-SW-82 Linux device driver allows the operation of the TIP840 Multi Channel ADC IndustryPack® modules conforming to the Linux I/O system specification. This includes a device-independent basic I/O interface with *open()*, *close()* and *ioctl()* functions.

Because the TIP840 device driver is stacked on the TEWS TECHNOLOGIES IPAC carrier driver, it's necessary to install also the appropriate IPAC carrier driver. Please refer to the IPAC carrier driver user manual for further information.

The TIP840-SW-82 device driver supports the following features:

- Convert and read data from input channel with selected gain and input interface
- Read module specific information (model-type and calibration data)

The TIP840-SW-82 device driver supports the modules listed below:

TIP840-10	8 Channel Single-Ended ADC (Gain: 1, 10, 100)	(IndustryPack®)
TIP840-11	8 Channel Single-Ended ADC (Gain: 1, 2, 4, 8)	(IndustryPack®)
TIP840-20	16 Channel Single-Ended (8 Channel Differential) ADC (Gain: 1, 10, 100)	(IndustryPack®)
TIP840-21	16 Channel Single-Ended (8 Channel Differential) ADC (Gain: 1, 2, 4, 8)	(IndustryPack®)

To get more information about the features and use of TIP840 devices it is recommended to read the manuals listed below.

TIP840 User manual

TIP840 Engineering Manual

CARRIER-SW-82 IPAC Carrier User Manual

1.1 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-82 is part of this TIP840-SW-82 distribution. It is located in directory CARRIER-SW-82 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-82 User Manual for a detailed description how to install and setup the CARRIER-SW-82 device driver, and for a description of the TEWS TECHNOLOGIES IPAC Carrier Driver concept.

2 Installation

Following files are located on the distribution media:

Directory path '.\TIP840-SW-82\':

TIP840-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
TIP840-SW-82-1.0.0.pdf	PDF copy of this manual
ChangeLog.txt	Release history
Release.txt	Release information

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

The GZIP compressed archive TIP840-SW-82-SRC.tar.gz contains the following files and directories:

Directory path './tip840/':

tip840drv.c	TIP840 device driver source
tip840def.h	TIP840 driver include file
tip840.h	TIP840 include file for driver and application
makenode	Script to create device nodes on the file system
Makefile	Device driver make file
example/tip840exa.c	Example application
example/Makefile	Example application make file
include/config.h	Driver independent library header file
include/tpmodule.h	Kernel independent library header file
include/tpmodule.c	Kernel independent library source code file

In order to perform an installation, extract all files of the archive TIP840-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TIP840-SW-82-SRC.tar.gz' will extract the files into the local directory.

- Login as *root* and change to the target directory
- Copy tip840.h to */usr/include*

Before building a new device driver, the TEWS TECHNOLOGIES IPAC carrier driver must be installed properly, because this driver includes the header file *ipac_carrier.h*, which is part of the IPAC carrier driver distribution. Please refer to the IPAC carrier driver user manual in the directory path *CARRIER-SW-82* on the distribution media.

2.1 Build and install the device driver

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

  # make install

  For Linux kernel 2.6.x, there may be compiler warnings claiming some undefined ipac_* symbols. These warnings are caused by the IPAC carrier driver, which is unknown during compilation of this TIP driver. The warnings can be ignored.

- Also after the first build we have to execute *depmod* to create a new dependency description for loadable kernel modules. This dependency file is later used by *modprobe* to automatically load the correct IPAC carrier driver modules.

  # depmod -aq

2.2 Uninstall the device driver

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

  # make uninstall

- Update kernel module dependency description file

  # depmod -aq

2.3 Install device driver into the running kernel

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

  # modprobe tip840drv

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

  # sh makenode

On success the device driver will create a minor device for each TIP840 module found. The first TIP840 can be accessed with device node /dev/tip840_0, the second TIP840 can be accessed with device node /dev/tip840_1 and so on.

The allocation of device nodes to physical TIP840 modules depends on the search order of the IPAC carrier driver. Please refer to the IPAC carrier user manual.

Loading of the TIP840 device driver will only work if kernel KMOD support is installed, necessary carrier board drivers already installed and the kernel dependency file is up to date. If KMOD support is not available you have to build either a new kernel with KMOD installed or you have to install the IPAC carrier kernel modules manually in the correct order (please refer to the IPAC carrier driver user manual).

2.4 Remove device driver from the running kernel

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

  # modprobe tip840drv -r

If your kernel has enabled a dynamic device file system all /dev/tip840_x nodes will be automatically removed from your file system after this.

Be sure that the driver is not opened by any application program. If opened you will get the response *´´tip840drv: Device or resource busy´´* and the driver will still remain in the system until you close all opened files and execute *modprobe -r* again.

2.5 Change Major Device Number

The TIP840 driver uses dynamic allocation of major device numbers by default. If this is not suitable for the application it is possible to define a major number for the driver. If the kernel has enabled a dynamic device file system the driver will not use the symbol TIP840_MAJOR.

To change the major number edit the file tip840.c, change the following symbol to appropriate value and enter **make install** to create a new driver.

TIP840_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TIP840_MAJOR       122
```

3 Device Input/Output functions

This chapter describes the interface to the device driver I/O system.

3.1 open()

NAME

open() - open a file descriptor

SYNOPSIS

#include <fcntl.h>

int open (const char *filename, int flags)

DESCRIPTION

The open function creates and returns a new file descriptor for the file named by *filename*. The *flags* argument controls how the file is to be opened. This is a bit mask; you create the value by the bitwise OR of the appropriate parameters (using the | operator in C). See also the GNU C Library documentation for more information about the open function and open flags.

EXAMPLE

```
int fd;

fd = open(“/dev/tip840_0”, O_RDWR);
if (fd < 0)
{
     /* handle open error */
}
```

RETURNS

The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

ENODEV	The requested minor device does not exist.

This is the only error code returned by the driver, other codes may be returned by the I/O system during open. For more information about open error codes, see the *GNU C Library description – Low-Level Input/Output*.

SEE ALSO

GNU C Library description – Low-Level Input/Output

3.2 close()

NAME

close() – close a file descriptor

SYNOPSIS

#include <unistd.h>

int close (int filedes)

DESCRIPTION

The close function closes the file descriptor *filedes*.

EXAMPLE

```
int fd;

if (close(fd) != 0)
{
     /* handle close error */
}
```

RETURNS

The normal return value from close is 0. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

ENODEV	The requested minor device does not exist.

This is the only error code returned by the driver, other codes may be returned by the I/O system during close. For more information about close error codes, see the *GNU C Library description – Low-Level Input/Output*.

SEE ALSO

GNU C Library description – Low-Level Input/Output

3.3 ioctl()

NAME

ioctl() – device control functions

SYNOPSIS

#include <sys/ioctl.h>
#include <tip840.h>

int ioctl(int filedes, int request [, void *argp])

DESCRIPTION

The **ioctl** function sends a control code directly to a device, specified by *filedes*, causing the corresponding device to perform the requested operation.

The argument *request* specifies the control code for the operation. The optional argument *argp* depends on the selected request and is described for each request in detail later in this chapter.

The following ioctl codes are defined in tip840.h:

Symbol	Meaning
TIP840_IOCG_READ	Start AD conversion and read value
TIP840_IOCG_GET_PARAM	Read module parameters

See behind for more detailed information on each control code.

RETURNS

On success, zero is returned. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

EINVAL	Invalid argument. This error code is returned if the requested ioctl function is unknown. Please check the argument request.

Other function dependent error codes will be described for each ioctl code separately. Note, the TIP840 driver always returns standard Linux error codes.

SEE ALSO

ioctl man pages

3.3.1 TIP840_IOCG_READ

NAME

TIP840_IOCG_READ – Start AD conversion and read input data

DESCRIPTION

This ioctl function starts an AD conversion and reads the converted value of a specified channel with specified parameters of the TIP840 associated with the open file descriptor, *filedes*, into the read data buffer pointed to by *argp*. Conversion parameters must be specified in the read data buffer.

The function automatically decides if the settling time has to pass or if just a simple conversion is needed. Therefore a conversion after a change of channel, gain or input interface will be slower than a conversion with unchanged parameters.

```
typedef struct
{
        unsigned int        channel;
        unsigned int        gain;
        unsigned int        mode;
        unsigned int        correction;
        int                 data;
} TIP840_READ_BUFFER, *PTIP840_READ_BUFFER;
```

channel

This value specifies the ADC input channel. Valid channel numbers depend on the installed model type. For TIP840-1x allowed channel numbers are 1…8, for TIP840-2x allowed channel numbers are 1…16 for single-ended input and 1…8 for differential input.

gain

This value specified the input gain. The following table shows the allowed gain values:

TIP840_GAIN_1	Select Gain 1x	(valid for all TIP840)
TIP840_GAIN_10	Select Gain 10x	(only valid for TIP840-x0)
TIP840_GAIN_100	Select Gain 100x	(only valid for TIP840-x0)
TIP840_GAIN_2	Select Gain 2x	(only valid for TIP840-x1)
TIP840_GAIN_4	Select Gain 4x	(only valid for TIP840-x1)
TIP840_GAIN_8	Select Gain 8x	(only valid for TIP840-x1)

mode

This value selects the channel input interface. If the input interface should be a differential interface, this member must be specified as *TIP840_DIFF*, otherwise the value should be *TIP840_SINGLE* for single-ended input. Differential interface is only available on TIP840-2x modules.

correction

This parameter selects if a data correction will be executed on the read value. If *TIP840_CORR* is specified, the input value will be corrected with factory set offset and gain correction data stored in the TIP840 ID-PROM. If the correction should not be executed, *TIP840_NOCORR* should be specified.

data

This field returns the converted and possibly corrected data value. The data is returned as a right aligned (12-bit) sign extended value. The range of returned values is between -2048 and +2047. The corresponding voltage depends on the selected gain. Please refer to the TIP840 hardware manual for further details.

EXAMPLE

```
include   <sys/ioctl.h>
include   <tip840.h>

int                     fd;
int                     result;
TIP840_READ_BUFFER      ReadBuf;

ReadBuf.channel    = 1;
ReadBuf.gain       = TIP840_GAIN_2;        /* Gain 2: -5V..+5V */
ReadBuf.mode       = TIP840_SINGLE;
ReadBuf.correction = TIP840_NOCORR;

result = ioctl(fd, TIP840_IOCG_READ, &ReadBuf);
if (result < 0)
{
     /* handle ioctl error */
}

printf("Input voltage = %8.5fV \n", (ReadBuf.data * 5.0) / 2048);
```

ERRORS

EFAULT	Invalid pointer to the read buffer.
EBUSY	Already a conversion active
ETIME	Conversion timed out

SEE ALSO

ioctl man pages

3.3.2 TIP840_IOCG_GET_PARAM

NAME

TIP840_IOCG_GET_PARAM - Read module parameters

DESCRIPTION

This ioctl function returns the module type and the calibration data of the TIP840 associated with the open file descriptor, *filedes*, into the parameter buffer pointed to by *argp.*

```
typedef struct
{
        unsigned int        model;
        int                 offset[4];
        int                 gain[4];
} TIP840_PARAM_BUFFER, *PTIP840_PARAM_BUFFER;
```

model

Receives the type code (10/11/20/21) of the associated TIP840.

offset

Receives the offset (zero) error of the input amplifier of the possible gain selections in the unit ¼ LSB (see also Hardware User Manual).

gain

Receives the gain error of the input amplifier of the possible gain selections in the unit ¼ LSB (see also Hardware User Manual).

EXAMPLE

```
include   <sys/ioctl.h>
include   <tip840.h>

int                     fd;
int                     result;
TIP840_PARAM_BUFFER     ParamBuf;

result = ioctl(fd, TIP840_IOCG_GET_PARAM, &ParamBuf);
if (result < 0)
{
     /* handle ioctl error */
}

printf("Read parameters from TIP840-%02d done\n", ParamBuf.model);
```

ERRORS

EFAULT	Invalid pointer to the read buffer.

SEE ALSO

ioctl man pages

4 Debugging

For debugging output see tip840.c. You will find the following symbol:

```
#undef TIP840_DEBUG_VIEW
```

To enable a debug output replace “undef” with “define”.

The TIP840_DEBUG_VIEW symbol controls debugging output from the whole driver.

```
TIP840 - 16 Channel 12 bit ADC version 1.0.0 (2009-02-26)
TIP840  :  Probe new TIP840 mounted on <TEWS TECHNOLOGIES - (Compact)PCI
IPAC Carrier> at slot D

TIP840  :  IP I/O Memory Space
00000000 : 3F FF FF FF F8 00 FF FF FF FF FF FF FF FF FF FF
00000010 : 3F FF FF FF F8 00 FF FF FF FF FF FF FF FF FF FF
00000020 : 3F FF FF FF F8 00 FF FF FF FF FF FF FF FF FF FF
00000030 : 3F FF FF FF F8 00 FF FF FF FF FF FF FF FF FF FF
00000040 : FF

TIP840  :  IP ID Memory Space
00000000 : FF 49 FF 50 FF 41 FF 43 FF B3 FF 0D FF 10 FF 00
00000010 : FF 00 FF 00 FF 15 FF EF FF 15 FF 00 FF 00 FF 00
00000020 : FF 00 FF 0C FF 0B FF 0B FF 0B FF 00 FF 00 FF 00
00000030 : FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00

TIP840  :  TIP840_IOCG_READ ioctl
TIP840  :  TIP840_IOCG_GET_PARAM ioctl
```