The Embedded I/O Company

TEWS TECHNOLOGIES

TIP501-SW-72

LynxOS Device Driver

16/8 Channel 16 Bit ADC

Version 2.0.x

User Manual

Issue 2.0.1

July 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

TIP501-SW-72

LynxOS Device Driver

16/8 Channel 16 Bit ADC

Supported Modules:
TIP501

Issue	Description	Date
1.0	First Issue	January 17, 2003
1.1	Corrections	January 27, 2003
2.0.0	General Revision, Carrier Support added	October 9, 2006
2.0.1	Carrier Driver description added	July 7, 2008

Table of Contents

1 Introduction

1.1 Device Driver

The TIP501-SW-72 LynxOS device driver allows the operation of a TIP501 IPAC module on LynxOS operating systems.

Because the TIP501 device driver is stacked on the TEWS TECHNOLOGIES IPAC carrier driver, it's necessary to install also the IPAC carrier driver. Please refer to the IPAC carrier driver user manual for further information.

The standard file (I/O) functions (open, close, read and ioctl) provide the basic interface for opening and closing a file descriptor and for performing device I/O and control operations.

The TIP501 device driver includes the following functions:

- Reading converted AD values from a specified analog input channel with or without data correction
- Reading module type and correction values stored in the ID PROM
- TEWS TECHNOLOGIES IPAC carrier driver support.

The TIP501-SW-72 supports the modules listed below:

TIP501-10	Optically isolated 16 channel 16 bit ADC input voltage range +/-10V, gain 1, 2, 5, 10	(IndustryPack ®)
TIP501-11	Optically isolated 16 channel 16 bit ADC input voltage range +/-10V, gain 1, 2, 4, 8	(IndustryPack ®)
TIP501-20	Optically isolated 16 channel 16 bit ADC input voltage range 0V to +10V, gain 1, 2, 5, 10	(IndustryPack ®)
TIP501-21	Optically isolated 16 channel 16 bit ADC input voltage range 0V to +10V, gain 1, 2, 4, 8	(IndustryPack ®)

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

TIP501 User manual

TIP501 Engineering Manual

CARRIER-SW-72 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-72 is part of this TIP501-SW-72 distribution. It is located in directory CARRIER-SW-72 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-72 User Manual for a detailed description how to install and setup the CARRIER-SW-72 device driver, and for a description of the TEWS TECHNOLOGIES IPAC Carrier Driver concept.

2 Installation

The directory TIP501-SW-72 on the distribution media contains the following files:

TIP501-SW-72-2.0.1.pdf	This manual in PDF format
TIP501-SW-72-SRC.tar	Device Driver and Example sources
ChangeLog.txt	Release history
Release.txt	Release information

The TAR archive TIP501-SW-72-SRC.tar contains the following files and directories:

Directory path 'tip501':

tip501.c	Driver source code
tip501.h	Definitions and data structures for driver and application
tip501def.h	Definitions and data structures for the driver
tip501_info.c	Device information definition
tip501_info.h	Device information definition header
tip501.cfg	Driver configuration file include
tip501.import	Linker imports file for PowerPC platforms
Makefile	Device driver make file
example/tip501exa.c	Example application source
example/Makefile	Example application make file

In order to perform a driver installation first extract the TAR file to a temporary directory then copy the following files to their target directories:

1. Create a new directory in the system drivers directory path /sys/drivers.xxx, where xxx represents the BSP that supports the target hardware.

   For example: /sys/drivers.pp_drm/tip501 or /sys/drivers.cpci_x86/tip501

2. Copy the following files to this directory:
   - tip501.c
   - tip501def.h
   - tip501.import
   - Makefile

3. Copy tip501.h to /usr/include/

4. Copy tip501_info.c to /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).

5. Copy tip501_info.h to /sys/dheaders/

6. Copy tip501.cfg to */sys/cfg.xxx/*, where xxx represents the BSP for the target platform

For example: /sys/cfg.ppc or /sys/cfg.x86 ....

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-72 on the separate distribution media.

2.1 Device Driver Installation

The two methods of driver installation are as follows:

- Static Installation
- Dynamic Installation (only native LynxOS systems)

Both installation methods require the TEWS TECHNOLOGIES IPAC Carrier Driver. Please refer to the IPAC Carrier Driver User Manual for detailed information.

2.1.1 Static Installation

With this method, the driver object code is linked with the kernel routines and is installed during system start-up.

2.1.1.1 Build the driver object

1. Change to the directory /sys/drivers.xxx/tip501, where xxx represents the BSP that supports the target hardware.
2. To update the library /sys/lib/libdrivers.a enter:

```
make install
```

2.1.1.2 Create Device Information Declaration

1. Change to the directory /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).
2. Add the following dependencies to the Makefile

```
DEVICE_FILES_all = ... tip501_info.x
```

   And at the end of the Makefile

```
tip501_info.o:$(DHEADERS)/tip501_info.h
```

3. To update the library /sys/lib/libdevices.a enter:

```
make install
```

2.1.1.3 Modify the Device and Driver Configuration File

In order to insert the driver object code into the kernel image, an appropriate entry in file CONFIG.TBL must be created.

1. Change to the directory /sys/lynx.os/ respective /sys/bsp.xxx, where xxx represents the BSP that supports the target hardware.
2. Create an entry at the end of the file CONFIG.TBL

   Insert the following entry at the end of this file. Be sure that the necessary TEWS TECHNOLOGIES IPAC carrier driver is included **before** this entry.

```
I:tip501.cfg
```

2.1.1.4 Rebuild the Kernel

1. Change to the directory /sys/lynx.os/ (/sys/bsp.xxx)
2. Enter the following command to rebuild the kernel:

   ```
   make install
   ```

3. Reboot the newly created operating system by the following command (not necessary for KDIs):

   ```
   reboot –aN
   ```

   The N flag instructs init to run mknod and create all the nodes mentioned in the new nodetab.

4. After reboot you should find the following new devices (depends on the device configuration): /dev/tip501_0, /dev/tip501_1, /dev/tip501_2, …

2.1.2 Dynamic Installation

This method allows you to install the driver after the operating system is booted. The driver object code is attached to the end of the kernel image and the operating system dynamically adds this driver to its internal structures. The driver can also be removed dynamically.

2.1.2.1 Build the driver object

1. Change to the directory /sys/drivers.xxx/tip501, where xxx represents the BSP that supports the target hardware.
2. To make the dynamic link-able driver enter :

```
make
```

2.1.2.2 Create Device Information Declaration

(1) Change to the directory /sys/drivers.xxx/tip501, where xxx represents the BSP that supports the target hardware.

(2) To create a device definition file for the major device (this work only on native system)

```
make t501info
```

(3) To install the driver enter:

```
drinstall –c tip501.obj
```

If successful, drinstall returns a unique <driver-ID>

(4) To install the major device enter:

```
devinstall –c –d <driver-ID> t501info
```

The <driver-ID> is returned by the drinstall command

(5) To create nodes for the devices enter:

```
mknod /dev/tip501_0 c <major_no> 0
mknod /dev/tip501_1 c <major_no> 1
mknod /dev/tip501_2 c <major_no> 2
…
```

The <major_no> is returned by the devinstall command.

If all steps are successful completed the TIP501 is ready to use.

2.1.2.3 Uninstall dynamic loaded driver

To uninstall the TIP501 device enter the following commands:

```
devinstall –u –c <device-ID>
drinstall –u <driver-ID>
```

2.1.3 Configuration File: CONFIG.TBL

The device and driver configuration file CONFIG.TBL contains entries for device drivers and its major and minor device declarations. Each time the system is rebuild, the config utility read this file and produces a new set of driver and device configuration tables and a corresponding nodetab.

To install the TIP501 driver and devices into the LynxOS system, the configuration include file tip501.cfg must be included in the CONFIG.TBL (see also 2.1.1.3).

The file tip501.cfg on the distribution disk contains the driver entry (*C:tip501:\....*) and a major device entry (*D:TIP501:t501info::*) with 9 minor device entries (*"N: tip501_0:0", ..., "N: tip501_8:8"*).

If the driver should support more than nine TIP501, additional minor device entries must be added. To create the device node */dev/tip501_9* the line *N:tip501_9:9* must be added at the end of the file tip501.cfg. For the next node a minor device entry with 10 must be added and so on.

This example shows the predefined driver entry:

```
#    Format:
#    C:driver-name:open:close:read:write:select:control:install:uninstall
#    D:device-name:info-block-name:raw-partner-name
#    N:node-name:minor-dev

C:tip501:\
     :t501open:t501close:t501read::\
     ::t501ioctl:t501install:t501uninstall
D:TIP501:t501info::
N:tip501_0:0
N:tip501_1:1
N:tip501_2:2
N:tip501_3:3
N:tip501_4:4
N:tip501_5:5
N:tip501_6:6
N:tip501_7:7
N:tip501_8:8
```

The configuration above creates the following nodes in the /dev directory.

/dev/tip501_0 … /dev/tip501_8

3 TIP501 Device Driver Programming

LynxOS system calls are all available directly to any C program. They are implemented as ordinary function calls to "glue" routines in the system library, which trap to the OS code.

Note that many system calls use data structures, which should be obtained in a program from appropriate header files. Necessary header files are listed with the system call synopsis.

3.1 open()

NAME

open() - open a file

SYNOPSIS

#include <sys/file.h>
#include <sys/types.h>
#include <fcntl.h>

int open (char *path, int oflags[, mode_t mode])

DESCRIPTION

Opens a file (TIP501 device) named in path for reading and writing. The value of oflags indicates the intended use of the file. In case of a TIP501 device oflags must be set to O_RDWR to open the file for both reading and writing.

The mode argument is required only when a file is created. Because a TIP501 device already exists this argument is ignored.

EXAMPLE

```
int fd

fd = open ("/dev/tip501_0", O_RDWR);
```

RETURNS

Open returns a file descriptor number if successful or -1 on error. The global variable *errno* contains the detailed error code.

3.2 close()

NAME

close() – close a file

SYNOPSIS

int close(int fd)

DESCRIPTION

This function closes an opened device associated with the valid file descriptor handle fd.

EXAMPLE

```
int fd
int result;

result = close(fd);
```

RETURNS

Close returns 0 (OK) if successful, or –1 on error. The global variable errno contains the detailed error code.

SEE ALSO

LynxOS System Call - close()

3.3 read()

NAME

read() - read from a file

SYNOPSIS

```
#include <tip501.h>

int read ( int fd, char *buff, int count )
```

DESCRIPTION

The read function attempts to start an AD conversion on the specified channel and returns the converted value in a read buffer to the caller.

A pointer to the callers read buffer (*T501_RW_BUFFER*) and the size of this structure is passed by the parameters *buff* and *count* to the device.

```
typedef struct
{
        unsigned int        channel;
        unsigned int        gain;
        unsigned int        mode;
        unsigned int        correction;
        long                data;
} T501_IO_BUFFER, *PT501_IO_BUFFER;
```

channel

Specifies the channel number at which to read the AD value. Valid channel numbers are 1...16 if Single-Ended is selected, if differential is selected the valid channel numbers are in the range of 1...8.

gain

Specifies the gain, which shall be used to read the AD value. Valid gains are:

T501_GAIN_1	Select Gain 1	Valid for TIP501-10/-11/-20/-21
T501_GAIN_2	Select Gain 2	Valid for TIP501-10/-11/-20/-21
T501_GAIN_4	Select Gain 4	Valid for TIP501-11/-21
T501_GAIN_5	Select Gain 5	Valid for TIP501-10/-20
T501_GAIN_8	Select Gain 8	Valid for TIP501-11/-21
T501_GAIN_10	Select Gain 10	Valid for TIP501-10/-20

mode

Specifies the channel input interface. If it should be used with a differential interface, this member must have the value *T501_DIFF*, otherwise the value should be *T501_SINGLE,* if it should be used with a single-ended input.

correction

If this parameter is *T501_CORR* the driver performs an automatic offset and gain correction with factory calibration data stored in the TIP501 ID-PROM, otherwise the value should be *T501_NOCORR*.

data

Analog input value read from the specified ADC channel. The analog data is returned as sign extended two's complement integer value with 16-bit resolution. The lower four bits are always 0 (see also TIP501 Hardware User Manual).

EXAMPLE

```
int fd;
int result;
T501_IO_BUFFER ADCBuf;

ADCBuf.gain        = T501_GAIN_1;
ADCBuf.mode        = T501_SINGLE;
ADCBuf.channel     = 1;
ADCBuf.correction  = T501_CORR;

result = read(fd, (char*)&ADCBuf, sizeof(ADCBuf));

if(result < 0)
{
     /* handle read error */
}
```

RETURNS

When read succeeds, the size of the read buffer is returned. If read fails, -1 (SYSERR) is returned.

On error, errno will contain a standard read error code (see also LynxOS System Call – read) or one of the following TIP501 specific error codes:

ENXIO	Illegal device
EINVAL	Invalid argument. This error code is returned if the size of the read buffer is too small or if the gain or channel parameter out of range.
EIO	AD conversion hasn't finished within the maximum allowed time period.
ETIMEDOUT	The fix device access timeout has elapsed because other read requests to this device are pending.
EAGAIN	You've set a timeout value, but there are no timeouts available. Do it again without a timeout.
EINTR	Interrupted system call (probably by a signal).

SEE ALSO

LynxOS System Call - read()

3.4 ioctl()

NAME

ioctl() - I/O device control

SYNOPSIS

```
#include <ioctl.h>
#include <tip501.h>

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

DESCRIPTION

ioctl provides a way of sending special commands to a device driver. The call sends the value of request and the pointer arg to the device associated with the descriptor fd.

The following ioctl codes are defined in tip501.h:

Value	Meaning
T501_READ_PARAM	Read module parameter

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.

The TIP501 ioctl function returns always standard error codes.

SEE ALSO

LynxOS System Call – ioctl() for detailed description of possible error codes.

3.4.1 T501_READ_PARAM

NAME

T501_READ_PARAM - Read module parameter

DESCRIPTION

This ioctl function attempts to read the module type and calibration data of the TIP501 associated with the open file descriptor *fd*, into the parameter buffer pointed to by *arg.*

The parameter buffer (*T501_PARAM_BUFFER*) has the following layout:

```
typedef struct
{
        unsigned int ModuleType;
        int          calGain[4];
        int          calOffs[4];
} T501_PARAM_BUFFER, *PT501_PARAM_BUFFER;
```

ModuleType

Receives the type code (10/11/20/21) of the associated TIP501.

calGain

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

calOffs

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

EXAMPLE

```
int fd;
int result;
T501_PARAM_BUFFER ParamBuf;

result = ioctl(fd, T501_READ_PARAM, &ParamBuf);

if (result < 0)
{
     /* handle ioctl error */
}
```

ERRORS

No function specific errors will be returned.

SEE ALSO

ioctl man pages

4 Debugging and Diagnostic

If your installed IPAC port driver (e.g. tip501) doesn't find any devices although the IPAC is properly plugged on a carrier port, it's interesting to know what's going on in the system.

Usually all TEWS TECHNOLOGIES device drivers announce significant event or errors via the device driver routine kkprintf(). To enable the debug output you must define the macro DEBUG in the device driver source files (e.g. carrier_class.c, carrier_tews_pci.c, tip501.c, ...).

The debug output should appear on the console. If not please check the symbol KKPF_PORT in uparam.h. This symbol should be configured to a valid COM port (e.g. SKDB_COM1).

The following output appears at the LynxOS debug console if the carrier and IPAC driver starts:

```
TIP501 - Optically Isolated 16 Channel 16 Bit ADC version 2.0.0 (2006-10-
09)
TIP501  :  Probe new TIP501 mounted on <TEWS TECHNOLOGIES - VME Carrier> at
slot B
TIP501  :  Probe new TIP501 mounted on <TEWS TECHNOLOGIES - (Compact)PCI
IPAC Carrier> at slot B
```

If you can't solve the problem by yourself, please contact TEWS TECHNOLOGIES with a detailed description of the error condition, your system configuration and the debug outputs.