The Embedded I/O Company

TEWS TECHNOLOGIES

TIP866-SW-72

LynxOS Device Driver

8-Channel Serial IPAC

Version 1.0.x

User Manual

Issue 1.0

December 2003

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

TEWS TECHNOLOGIES LLC
1 E. Liberty Street, Sixth Floor
Reno, Nevada 89504 / USA
Phone: +1 (775) 686 6077
Fax: +1 (775) 686 6024
e-mail: usasales@tews.com
www.tews.com

TIP866-SW-72

8-Channel Serial IPAC

LynxOS Device Driver

Issue	Description	Date
1.0	First Issue	December 5, 2003

Table of Contents

1 Introduction

The TIP866-SW-72 LynxOS device driver is a full-duplex tty device driver, which allows the operation of a TIP866 serial IPAC module on LynxOS operating systems.

Because the TIP866 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 TIP866-SW-72 device driver is based on the standard LynxOS terminal manager. Due to this way of implementation the driver interface and function is absolute compatible to the standard LynxOS terminal driver.

All standard utility programs for configuration and maintaining terminal interfaces could be used in the same manner. There is only one exception; the TIP866-SW-72 can't be used as console driver.

Additional supported features:

- Baud rates up to 115200 baud for TIP866-10 and up to 460800 baud for TIP866-11 and TIP866-20.
- Direct support of different physical interfaces (RS-232, TTL and RS-422).
- Each channel has a 64 Byte transmit and receive FIFO
- Programmable trigger level for transmit and receive FIFO.
- Software handshake (Xon/Xoff) direct controlled by the serial controller. The advantage of this feature is that the transmission of characters will immediately stop as soon as a complete character is transmitted and not when the transmit FIFO is empty for handshake under software control.
- Static and dynamic installation support.
- TEWS TECHNOLOGIES IPAC carrier driver support.

2 Installation

The software is delivered on a PC formatted 3½" HD diskette.

The directory A:\TIP866-SW-72 contains the following files:

TIP866-SW-72.pdf	This manual in PDF format
TIP866-SW-72.tar	Device driver sources

The TAR archive TIP866-SW-72.tar contains the following files and directories:

tip866/tip866.c	Driver source code
tip866/tip866.h	Driver and application include file
tip866/tip866_info.c	Device information definition
tip866/tip866_info.h	Device information definition header
tip866/tip866.cfg	Driver configuration file include
tip866.import	Linker import file for Power PC platforms
tip866/Makefile	Device driver make file for static installation
tip866/Example/example.c	Example application source
tip866/Example/Makefile	Example 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/tip866 or /sys/drivers.cpci_x86/tip866

2. Copy the following files to this directory:
   - tip866.c
   - tip866.import
   - Makefile

3. Copy tip866.h to /usr/include/

4. Copy tip866_info.c to /sys/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).

5. Copy tip866_info.h to /sys/dheaders/

6. Copy tip866.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 *A:\CARRIER-SW-72* on the separate distribution diskette.

2.1 Device Driver Installation

The two methods of driver installation are as follows:

- Static Installation
- Dynamic Installation

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/tip866, 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 = ... tip866_info.x
   ```

   And at the end of the Makefile

   ```
   tip866_info.o:$(DHEADERS)/tip866_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:tip866.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.

After reboot you should find the following new devices (depends on the device configuration): */dev/tip866_0, [/dev/tip866_15, …]*

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/tip866, 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/devices.xxx/ or /sys/devices if /sys/devices.xxx does not exist (xxx represents the BSP).
2. To create a device definition file for the major device (this works only on native system)

   ```
   make t866info
   ```

3. To install the driver enter:

   ```
   drinstall –c tip866.obj
   ```

   If successful, drinstall returns a unique <driver-ID>
4. To install the major device enter:

   ```
   devinstall –c –d <driver-ID> t866info
   ```

   The <driver-ID> is returned by the drinstall command
5. To create nodes for the devices enter:

   ```
   mknod /dev/tip866_0 c <major_no> 0
   mknod /dev/tip866_1 c <major_no> 1
   mknod /dev/tip866_2 c <major_no> 2
   ...
   ```

   The <major_no> is returned by the devinstall command.

If all steps are successful completed the TIP866 is ready to use.

2.1.2.3 Uninstall dynamic loaded driver

To uninstall the TIP866 device enter the following commands:

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

2.1.3 Device Information Definition File

The device information definition contains information necessary to install the TIP866 major device.

The implementation of the device information definition is done by a C structure which is defined in the header file tip866_info.h.

This structure contains following parameters:

FIFO_Settings

Contains the trigger level for receive and transmit FIFO's.
Valid settings for the receiver FIFO are: 1, 8, 16, 56 and 60.
Valid settings for the transmitter FIFO are: 1, 8, 16, 32 and 56.
NOTE. A trigger level of 1 will disable the receive or transmit FIFO.

sg

This structure contains initial tty parameter like baud rate special characters and so on. Refer also to the tty man pages.

The settings above will be used as default settings for all minor devices (serial channels) in common.

A TIP866 major device can support an "unlimited" number of minor devices respective TIP866 IPAC modules. The order in which the TIP866 channels are assigned to the device nodes in the /dev directory depends on the search order of the TEWS TECHNOLOGIES IPAC carrier driver. Please refer to the IPAC Carrier driver manual for further information.

If only one TIP866 is used the device node /dev/tip866_0 corresponds to the serial channel 1, node /dev/tip866_1 to serial channel 2 and so on.

```
T866_INFO t866info = {
     { 56, 8 },                          /* Rx, Tx trigger level          */

     {
          B9600, B9600,                  /* input and output speed        */
          'H' - '@',                     /* erase char                    */
          -1,                            /* 2nd erase char                */
          'U' - '@',                     /* kill char                     */
          ECHO | CRMOD,                  /* mode                          */
          'C' - '@',                     /* interrupt character           */
          '\\' - '@',                    /* quit char                     */
          'Q' - '@',                     /* start char                    */
          'S' - '@',                     /* stop char                     */
          'D' - '@',                     /* EOF                           */
          -1,                            /* brk                           */
          (LCRTBS | LCRTERA | LCRTKIL | LCTLECH), /* local mode word      */
          'Z' - '@',                     /* process stop                  */
          'Y' - '@',                     /* delayed stop                  */
          'R' - '@',                     /* reprint line                  */
          'O' - '@',                     /* flush output                  */
          'W' - '@',                     /* word erase                    */
          'V' - '@'                      /* literal next char             */
     }
};
```

2.1.4 Configuration File: CONFIG.TBL

The device and driver configuration file CONFIG.TBL contain 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 TIP866 driver and devices into the LynxOS system, the configuration include file tip866.cfg must be included in the CONFIG.TBL (see also 2.1.1.3).

The file tip866.cfg on the distribution disk contains the driver entry (C:tip866:\....) and a major device entry (D:TIP866:t866info::) with 16 minor device entries (N: tip866_0:0 ... N:tip866_15:15).

If the driver should support more than 16 minor devices (serial channels) because more than 2 TIP866 are plugged, additional minor device entries must be added. To create the device node /dev/tip866_16 the line N:tip866_16:16 must be added at the end of the file tip866.cfg. For the next node a minor device entry with 17 must be added and so on.

NOTE. The name of the device information declaration (info-block-name) must match to an existing C structure in the file tip866_info.c.

This example shows a driver entry with a major device and 16 minor devices:

```
#    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:tip866:\
     :t866open:t866close:t866read:t866write:\
     :t866select:t866ioctl:t866install:t866uninstall
D:TIP866:t866info::
N:tip866_0:0
N:tip866_1:1
...
N:tip866_15:15


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

/dev/tip866_0
/dev/tip866_1
...
/dev/tip866_15
```

3 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.

Because the TIP866-SW-72 device driver is based on the standard LynxOS terminal manager, the driver interface and functions are absolute compatible to the standard LynxOS terminal driver and support System V and POSIX terminal programming.

Some function were added to provide configuration of hardware handshake (only TIP866-10 and TIP866-11) and programming of the trigger level of the transmit and receive FIFO if different settings on serial channels are necessary. The default configuration of all channels for the trigger levels of the transmit and receive FIFO can be changed by modifying the device information file tip866_info.c (see also 2.1.3).

3.1 ioctl()

NAME

ioctl() – device control functions

SYNOPSIS

```
#include <ioctl.h>
#include <tip866.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 argument *request* specifies the control code for the operation. The optional argument *arg* depends on the selected request and is described for each request in detail later in this chapter.

The following additional ioctl commands are supported:

Symbol	Meaning
TIOSHWH	Enable hardware handshake with RTS/CTS
TIOCHWH	Disable hardware handshake
TIOSFIFO	Set FIFO trigger levels

See behind for more detailed information on each control code.

To use these ioctl commands the header file tip866.h must be included.

3.1.1 TIOSHWH

NAME

TIOSHWH - Enable hardware handshake with RTS/CTS

DESCRIPTION

This *ioctl()* function enables hardware handshake with RTS/CTS (not supported for TIP866-20) on the specified minor device (serial channel).

The optional argument isn't used and can be set to 0.

EXAMPLE

```
int fd;
int result;


fd = open( "/dev/tip866_0", O_RDWR );


/* enable hardware handshake */
result = ioctl( fd, TIOSHWH, 0 );
```

ERRORS

EPERM	Operation not permitted. This error is returned if the hardware handshake should be enabled on TIP866-20 modules.

SEE ALSO

ioctl man pages

3.1.2 TIOCHWH

NAME

TIOCHWH - Disable hardware handshake

DESCRIPTION

This *ioctl()* function disables hardware handshake with RTS/CTS (not supported for TIP866-20) on the specified minor device (serial channel).

The optional argument isn't used and can be set to 0.

EXAMPLE

```
int fd;
int result;


fd = open( "/dev/tip866_0", O_RDWR );


/* disable hardware handshake */
result = ioctl( fd, TIOCHWH, 0 );
```

ERRORS

EPERM	Operation not permitted. This error is returned if the hardware handshake should be disabled on TIP866-20 modules.

SEE ALSO

ioctl man pages

3.1.3 TIOSFIFO

NAME

TIOSFIFO - Set new FIFO trigger level

DESCRIPTION

This *ioctl()* function setup the receive and transmit FIFO trigger level on the specified minor device (serial channel). Calling this ioctl() function is only necessary if the default trigger level configuration (RX:56/TX:8) isn't suitable for a certain serial channels. Otherwise the default trigger level in the device information file should be adapted (see also 2.1.3).

A pointer to the callers buffer (*T866_FIFO_SETTINGS*) is passed by the parameter *arg* to the driver.

The *T866_FIFO_SETTINGS* structure has the following layout:

```
typedef struct {
        unsigned short          RxTriggerLevel;
        unsigned short          TxTriggerLevel;
} T866_FIFO_SETTINGS;
```

RxTriggerLevel

Contains the trigger level for the receive FIFO.
Valid settings are: 1, 8, 16, 56 and 60.

TxTriggerLevel

Contains the trigger level for the transmit FIFO.
Valid settings are: 1, 8, 16, 32 and 56.

This function should be called only if no data transfer is in progress, otherwise data will be lost.

EXAMPLE

```
int  fd;
int  result;
T866_FIFO_SETTINGS  fifo;


fd = open( “/dev/tip866_0”, O_RDWR );


/* setup new FIFO trigger level */
fifo.RxTriggerLevel = 60;
fifo.TxTriggerLevel = 16;


result = ioctl( fd, TIOSFIFO, &fifo );
```

ERRORS

EINVAL	Invalid argument. At least one trigger level was out of range.

SEE ALSO

ioctl man pages

4 Diagnostic

If your installed IPAC port driver (e.g. tip866) 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 driver announced 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, tip866.c,...).

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

```
TEWS TECHNOLOGIES - IPAC Carrier Class Driver version 1.0.0 (2003-11-28)
TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier version 1.0.0 (2003-11-28)
TEWS_PCI :  Probe new device (vendor=0x1498, device=0x30C8, #slots=4)
IPAC_CC  :  IPAC (Manuf-ID=B3, Model#=1D) recognized @ slot=3 carrier=<TEWS TECHNOLOGIES -
(Compact)PCI IPAC Carrier>

TIP866 - 8 Channel Serial IP version 1.0.0 (2003-12-04)
TIP866 : Probe new TIP866 mounted on <TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier> at slot D
```

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.