The Embedded I/O Company

TEWS TECHNOLOGIES

TPCI868-SW-72

LynxOS Device Driver

16 Channel RS232

User Manual

Issue 1.0 Version 1.0.0

February 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

TPCI868-SW-72

16 Channel RS232

LynxOS Device Driver

Issue	Description	Date
1.0	First Issue	February 25, 2003

Table of Content

1 Introduction

The TPCI868-SW-72 LynxOS device driver is a full-duplex tty device driver which allows the operation of a TPCI868 serial PMC on a PowerPC platform with DRM based PCI interface.

The TPCI868-SW-72 device driver bases 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 TPCI868-SW-72 can't be used as console driver.

Additional supported features:

- Extended baudrates up to 921.6 kBaud.
- Each channel has a 64 Byte transmit and receive FIFO
- Programmable trigger level for transmit and receive FIFO.
- Hardware (RTS/CTS) and 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.

2 Installation

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

Following files are located on the diskette:

tpci868.c	Driver source code
tpci868_info.c	Device information definition
tpci868_info.h	Device information definition header
tpci868.cfg	Driver configuration file include
tpci868.import	Linker import file
Makefile	Device driver make file
Makefile.dldd	Make file for dynamic driver installation
tpci868-SW-72.pdf	This Manual in PDF format

2.1 Device Driver Installation

The two methods of driver installation are as follows:

- Static Installation
- Dynamic Installation

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.

In order to perform a static installation, copy the following files to the target directories:

1. Create a new directory in the system drivers directory path.

   For example: */sys/drivers.pp_drm/tpci868*

2. Copy the following files to this directory: *tpci868.c, Makefile*
3. Copy *tpci868_info.c* to */sys/devices/*
4. Copy *tpci868_info.h* to */sys/dheaders/*
5. Copy *tpci868.cfg* to */sys/cfg.ppc/*

2.1.1.1 Build the driver object

Change to the directory /sys/drivers.pp_drm/tpci868

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/*
2. Add the following dependencies to the *Makefile*

   ```
   DEVICE_FILES_prep = ...tpci868_info.x
   ```

3. And at the end of the Makefile

   ```
   tpci868_info.o:$(DHEADERS)/tpci868_info.h
   ```

4. 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/
2. Create an entry in the file CONFIG.TBL

   Insert the entry after the console driver section

   ```
   # End of console devices
   I:tpci868.cfg
   ```

2.1.1.4 Rebuild the Kernel

1. Change to the directory /sys/lynx.os/
2. To rebuild the kernel enter the following command:

   ```
   make install
   ```

3. Reboot the newly-created operating system by the following command:

   ```
   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/tpci868a1 ... /dev/tpci868a16*

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 it's internal structures. The driver can also be removed dynamically.

Unlike the description of the dynamic installation in the manual "Writing Device Drivers for LynxOS", the driver source must be placed in a directory under */sys/drivers.pp_drm/*

The following steps describe how to do a dynamic installation:

1. Create a new directory in the system drivers directory path.

   For example: */sys/drivers.pp_drm/tpci868*

2. Copy the following files to this directory:

   - tpci868.c
   - tpci868_info.c
   - tpci868_info.h
   - tpci868.import
   - Makefile.dldd

3. Change to the directory */sys/drivers.pp_drm/tpci868*

4. To make the dynamic link-able driver enter :

   ```
   make –f Makefile.dldd
   ```

5. Create a device definition file for one major device

   ```
   gcc –DDLDD –o tpci868_info tpci868_info.c
   ./tpci868_info > tpci868A
   ```

6. To install the driver enter:

   ```
   drinstall –c tpci868.obj
   ```

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

7. To install the major device enter:

   ```
   devinstall –c –d <driver-ID> tpci868A
   ```

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

8. To create nodes for the devices enter:

   mknod /dev/tpci868a1 c <*major_no*> 0

   mknod /dev/tpci868a2 c <*major_no*> 1

   ...

If all steps are successfully completed the TPCI868 is ready to use.

To uninstall the TPCI868 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 TPCI868 major device.

The implementation of the device information definition is done through a C structure which is defined in the header file *tpci868_info.h*.

This structure contains following parameter:

PCIBusNumber

Contains the PCI bus number at which the TPCI868 is connected. Valid bus numbers are in range from 0 to 255.

PCIDeviceNumber

Contains the device number (slot) at which the TPCI868 is connected. Valid device numbers are in range from 0 to 31.

If both *PCIBusNumber* and *PCIDeviceNumber* are –1 then the driver will autoscan for the TPCI868 device. The first device found in the scan order will be allocated by the driver for this major device. Already allocated devices can't be allocated twice. This is important to know if you have more that one TMPC868 major device.

FIFO_Settings[]

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

A trigger level of 1 will disable receive or transmit FIFO.

Sg

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

A device information definition is unique for every TPCI868 major device. The file *tpci868_info.c* on the distribution disk contains two device information declarations, **tpci868A** for the first major device and **tpci868B** for the second major device.

If the driver should support more than two major devices it is necessary to copy and paste an existing declaration and rename it with unique name for example **tpci868C**, **tpci868D** and so on.

It is also necessary to modify the device and driver configuration file, respectively the configuration include file *tpci868.cfg*.

The following device declaration information uses the auto find method to detect the TPCI868 module on PCI bus.

The Rx trigger level is set to 56 Byte and the Tx trigger level is set to 4 for all minor devices.

```
TPCI868_INFO tpci868A = {
     -1,                    /* auto find the TPCI868 on any PCI bus   */
     -1,

     {                      /* Rx, Tx trigger level                   */
          { 56, 8 },        /* channel 1  ( minor device 0 )          */
          { 56, 8 },        /* channel 2  ( minor device 1 )          */
          { 56, 8 },        /* channel 3  ( minor device 2 )          */
          { 56, 8 },        /* channel 4  ( minor device 3 )          */
          { 56, 8 },        /* channel 5  ( minor device 4 )          */
          { 56, 8 },        /* channel 6  ( minor device 5 )          */
          { 56, 8 },        /* channel 7  ( minor device 6 )          */
          { 56, 8 },        /* channel 8  ( minor device 7 )          */
          { 56, 8 },        /* channel 9  ( minor device 8 )          */
          { 56, 8 },        /* channel 10 ( minor device 9 )          */
          { 56, 8 },        /* channel 11 ( minor device 10 )         */
          { 56, 8 },        /* channel 12 ( minor device 11 )         */
          { 56, 8 },        /* channel 13 ( minor device 12 )         */
          { 56, 8 },        /* channel 14 ( minor device 13 )         */
          { 56, 8 },        /* channel 15 ( minor device 14 )         */
          { 56, 8 },        /* channel 16 ( minor device 15 )         */
     },

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

To install the TPCI868 driver and devices into the LynxOS system, the configuration include file *tpci868.cfg* must be included in the *CONFIG.TBL* (see also 2.1.1.3).

The file *tpci868.cfg* on the distribution disk contains the driver entry (C:tpci868:\....) and one enabled major device entry (D:TPCI868 1-16:tpci868A::) with 4 minor device entries (N: tpci868a1:0 ... N:tpci868a16:15).

If the driver should support more than one major device the following entries for major and minor devices must be enabled by removing the comment character (#). By copy and paste an existing major and minor entry and renaming the new entries, it is possible to add any number of additional TPCI868 device.

The name of the device information declaration (info-block-name) must match to an existing C structure in the file *tpci868_info.c*.

This example shows a driver entry with one 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:tpci868:\
     :tpci868open:tpci868close:tpci868read:tpci868write:\
     :tpci868select:tpci868ioctl:tpci868install:tpci868uninstall
D:TPCI868 1-16:tpci868A::
N:tpci868a1:0
N:tpci868a2:1
N:tpci868a3:2
N:tpci868a4:3
N:tpci868a5:4
N:tpci868a6:5
N:tpci868a7:6
N:tpci868a8:7
N:tpci868a9:8
N:tpci868a10:9
N:tpci868a11:10
N:tpci868a12:11
N:tpci868a13:12
N:tpci868a14:13
N:tpci868a15:14
N:tpci868a16:15
```

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

```
/dev/tpci868a1
/dev/tpci868a2
/dev/tpci868a3
/dev/tpci868a4
/dev/tpci868a5
/dev/tpci868a6
/dev/tpci868a7
/dev/tpci868a8
/dev/tpci868a9
/dev/tpci868a10
/dev/tpci868a11
/dev/tpci868a12
/dev/tpci868a13
/dev/tpci868a14
/dev/tpci868a15
/dev/tpci868a16
```

3 TPCI868 Device Driver Programming

The TPCI868-SW-72 device driver bases 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.

The TPCI868-SW-72 device driver supports two additional ioctl functions to enable and disable the hardware handshake (RTS/CTS).

TIOSHWH (290)* Enable the hardware handshake with RTS/CTS. This functions is only accepted for TPCI868-10 modules, all other TPCI868 variants returns **SYSERR**.

TIOCHWH (291)* Disable the hardware handshake.

* Function code

No arguments are necessary for these commands. The parameter *arg* in the device control function call can be set to 0.

Example (without checking error situations!):

```
#define TIOSHWH    290
#define TIOCHWH    291

main()
{
     int fd;
     int result;

     /* open the desired TPCI868 device */
     fd = open( "/dev/tpci868a1", O_RDWR );

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

     /* ... do I/O ... */

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

     close( fd );
}
```