The Embedded I/O Company

TEWS TECHNOLOGIES

CARRIER-SW-82

Linux Device Driver

IPAC Carrier

Version 1.3.x

User Manual

Issue 1.3.1

December 2006

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

CARRIER-SW-82

IPAC Carrier Device Driver

Issue	Description	Date
1.0	First Issue	October 2, 2002
1.1	New driver naming convention	January 30, 2003
1.2	VMEbus IPAC carrier support	March 5, 2004
1.3	Reject specific Carrier Boards	September 23, 2004
1.1.4	Issue format changed	November 3, 2004
1.2.0	Kernel 2.6.x support	November 30, 2004
1.2.1	Modified install description and file list	February 13, 2006
1.3.0	Kernel 2.6.x VMEbus support, /proc interface	June 7, 2006
1.3.1	Distribution file list and address of TEWS LLC modified	December 21, 2006

Table of Contents

1 Introduction

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 driver which work with any supported carrier board, TEWS TECHNOLOGIES has designed a software architecture that hides all of these carrier board differences under a well defined interface.

Basically the IPAC and carrier device drivers are implemented with a three level module stacking. The carrier port driver is the lowest level. It handles the implementation details of the IPAC carrier board. The carrier class driver at the second level includes the management of IPAC slots and modules and provides a common interface between the IPAC driver and the carrier board driver. At the highest level resides the IPAC port driver.

Other benefits of this software architecture are the hot-plugging and Plug and Play facility. After installation of the required device drivers and loading the carrier class driver, this driver will recognize supported carrier boards by itself. He will start the required carrier port drivers; collect information about plugged IPAC modules and starts appropriate IPAC port drivers.

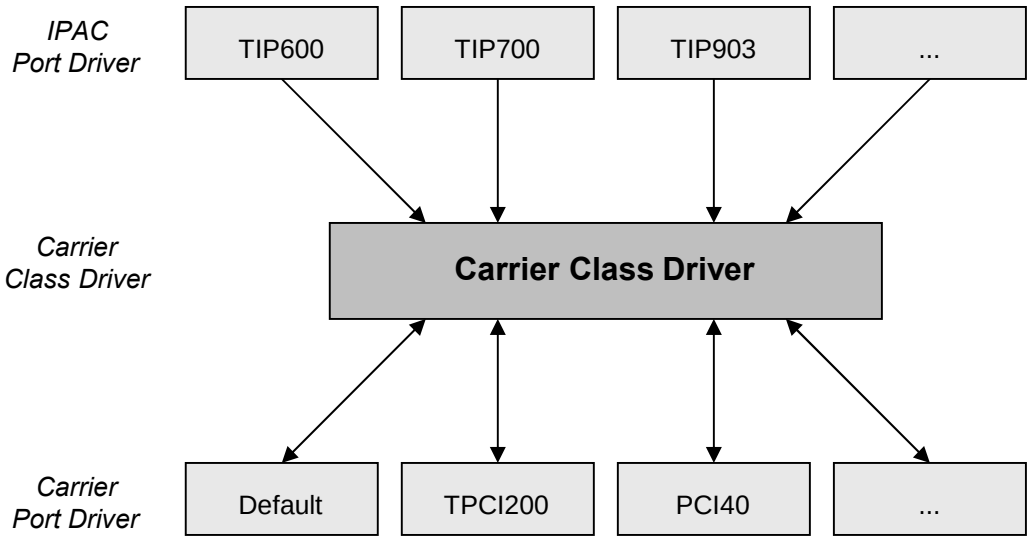

Figure 1: Stacked Driver Architecture

2 Installation

Usually the software is delivered together with the IPAC port driver.

The directory CARRIER-SW-82 on the distribution media contains the following files and directories:

CARRIER-SW-82-1.3.0.pdf	This manual in PDF format
CARRIER-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
unisdk-patch.tar.gz	GZIP compressed archive with SBS UniSDK patches
ChangeLog.txt	Release history
Release.txt	Release information

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

Directory path './ipac_carrier/':

ipac_carrier.h	Common used include file
class	Sub-directory with carrier class driver sources
default	Sub-directory with default carrier port driver sources
tews_pci	Sub-directory with TEWS PCI carrier port driver sources
sbs_pci	Sub-directory with SBS PCI carrier port driver sources
vme	Sub-directory with VME carrier port driver files
include	Sub-directory with driver independent library functions

In order to perform an installation, extract all files of the archive CARRIER-SW-82-SRC.tar.gz to the desired target directory.

The common used include file *ipac_carrier.h* must be copied also into the standard Linux include directory to */lib/modules/<kernel-version>/build/include* and */usr/include* to be available for installed IPAC port drivers. This file also provides the list of rejected PCI devices.

2.1 Build and install carrier drivers

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

  # make install

- Change to the appropriate sub-directory for your carrier board (e.g. /tews_pci if the IPAC modules are plugged on a TPCI200 carrier board)
- To create and install the driver in the module directory */lib/modules/<kernel-version>/misc* enter:

  # make install

- 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 dependent kernel modules.

  # depmod –aq

2.2 Uninstall the device driver

- Login as *root*
- Change to the desired carrier driver sub-directory.
- To remove the driver from the module directory */lib/modules/<kernel-version>/misc* enter:

  # make uninstall

- Update kernel module dependency description file

  # depmod –aq

2.3 Install the device driver in the running kernel

If KMOD support is available (should be standard for most of all Linux distributions) and all module dependencies are known (depmod) it's only necessary to load the carrier class driver with:

 # modprobe carrier_class

The carrier class driver will check the entire PCI bus for known IPAC carrier boards and starts the appropriate carrier port drivers (e.g. carrier_tews_pci). Loaded carrier port drivers will announce their resources (IPAC slots) to the carrier class driver. The carrier class driver checks each IPAC slot for plugged modules and starts the appropriate IPAC port drivers (e.g. tip903drv) if necessary.

In this scenario, it's not necessary to start any other device driver manually except the carrier class driver.

If this automatic start mechanism isn't desired the macro *CARRIER_PnP* in ./class/carrier_class.c must be removed (#undef).

Because all driver (module) dependencies are known, it's also possible to start the IPAC port driver (e.g. tip903drv) or the carrier port driver (e.g. carrier_tews_pci) first. All dependent drivers will be start automatically by modprobe or the carrier class driver.

The following screen shot shows the installed drivers and their dependencies:

```
# cat /proc/modules
tip903drv               8936   0 (autoclean) (unused)
carrier_tews_pci        5544   1 (autoclean)
carrier_class          10692   3 [tip903drv carrier_tews_pci]
```

If KMOD support isn't 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.

Assuming a TIP903 is plugged on a TPCI200 carrier board, you have to install necessary driver in the following order:

 # modprobe carrier_class

 # modprobe carrier_tews_pci

 # modprobe tip903drv

The carrier class driver must be always the first. The order of all other drivers doesn't matter.

2.4 Remove device driver from the running kernel

Removing of IPAC port, carrier class and carrier port drivers must be done in the following order:

- IPAC port driver (e.g. tip816drv)

  # modprobe tip816drv -r

- Carrier port driver (e.g. carrier_tews_pci)

  # modprobe carrier_tews_pci -r

- Carrier class driver (carrier_class)

  # modprobe carrier_class -r

Be sure that the driver isn't opened by any application program. If opened you will got the response "*tip816drv: Device or resource busy*" and the driver will still remain in the system until you close all opened files and execute *modprobe <module> –r* again.

3 VMEbus IPAC Carrier

3.1 UniSDK Patch

The current VMEbus IPAC carrier driver supports only VMEbus access via a modified SBS UniSDK driver for Intel x86 platforms. The symbiosis with the SBS UniSDK driver allows concurrent access to the VMEbus from UniSDK based applications and IPAC carrier based drivers.

Due to the fact that the UniSDK driver is distributed as binary, the already installed driver must be replaced by our modified UniSDK driver. The GZIP compressed tar archive unisdk-patch.tar.gz on the distribution media contains driver patches for UniSDK V3.4 (RedHat Linux 7.1) and UniSDK V4.1 (SUSE 10.0). For installation extract the tar archive to a temporary folder and copy the driver files to the appropriate target directory and restart the UniSDK driver or the entire system to make the changes current.

3.1.1 UniSDK 3.4 - RedHat Linux 7.1

Search for the file vmedrv.o in the lib/modules directory path (usually /lib/modules/2.4.18-/misc) and replace this file with ./UniSDK/3.4/vmedrv.o from the tar archive.

3.1.2 UniSDK 4.1 - SUSE 10.0

Search for the file sbs-unisdk.ko in the /lib/modules directory path (usually /lib/modules/2.6.13-15-default/extra/) and replace this file with ./UniSDK/4.1/sbs-unisdk.ko from the tar archive.

3.2 Configuration

After loading the new UniSDK VMEbus driver into the kernel the VMEbus IPAC carrier driver must be configured. Due to the fact that the VMEbus isn't a Plug&Play bus, VMEbus resources (memory, interrupts, etc.) must be configured manually. The header file *resource.h* in the *"/ipac_carrier/vme"* directory contains two tables for setting up required VMEbus memory windows (*image_desc[]*) and for declaring used IPAC carrier slots (*slot_desc[]*). All table entries must correspond to the real VMEbus carrier setup done by rotary switches or simple jumper configuration.

The default configuration in resource.h, setup two VMEbus windows (A16/D16 and A24/D16).

```
{  A16D16, 0x00000000, 0x00010000,  VME_A16,  VME_D16,  0,  -1 },
{  A24D16, 0x00D00000, 0x00100000,  VME_A24,  VME_D16,  0,  -1 },
```

The VMEbus window setup and the following IPAC slot setup are valid for the factory (default) setup of the TEWS TECHNOLOGIES VMEbus carrier TVME200.

```
{  0, 0x00006080,  0x80, A16D16, 0x00006000,  0x80, A16D16, 0x00D00000,
0x040000, A24D16, 64, 64,  1, 2 },
{  1, 0x00006180,  0x80, A16D16, 0x00006100,  0x80, A16D16, 0x00D40000,
0x040000, A24D16, 68, 68,  3, 4 },
{  2, 0x00006280,  0x80, A16D16, 0x00006200,  0x80, A16D16, 0x00D80000,
0x040000, A24D16, 72, 72,  5, 6 },
{  3, 0x00006380,  0x80, A16D16, 0x00006300,  0x80, A16D16, 0x00DC0000,
0x040000, A24D16, 76, 76,  7, 0 },
```

If the default configuration isn't suitable the existing entries can be modified as required. New entries can be added at the end of the list. Please refer to the comments in resource.h for detailed description of each parameter.

To make the new configuration current please rebuild the driver by entering *make install*. If the *vme* driver is already installed remove the driver from the running kernel first (see also 2.4) and install the new *vme* driver (see also 2.3).

4 Customer IPAC Carrier Support

If your IPAC carrier isn't supported by the carrier port drivers on the distribution diskette and your carrier board is a PCI bus carrier please contact TEWS TECHNOLOGIES.

Usually we will implement the carrier port driver without any charge within a few days.

If your carrier board doesn't require any initialization or special interrupt or error handling you can create IPAC slot entries in the default carrier port driver. The default carrier port driver will be loaded automatically by the carrier class driver.

To add IPAC slots you must change to the sub-directory ./default in the installation directory. Open the source file carrier_default.c in an appropriate editor and add a new entry in the array slot_info[] after the comment /* Please add slot entries here! */.

The creation of a new slot entry is very easy. Please copy and paste an entry from the example and change address and interrupt parameter as necessary. Be sure using always physical addresses! All fields are described in detail in the structure definition above.

You must create a slot entry for each slot. If you have carrier board with four slots you have to create four slot entries.

After modification you have to build and install the default driver like any other carrier port driver (see also 2.1).

5 Appendix

5.1 Supported IPAC Carrier Boards

The following TEWS TECHNOLOGIES and SBS IPAC carrier boards are supported:

Driver	Carrier Board	Descrition
carrier_tews_pci	TPCI100	PCI carrier for 2 IndustryPack modules
	TPCI200	PCI carrier for 4 IndustryPack modules
	TCP201	Compact PCI carrier for 4 IndustryPack modules
	TCP211	Compact PCI carrier for 2 IndustryPack modules
	TCP212	Compact PCI carrier for 2 IndustryPack modules
	TCP213	Compact PCI carrier for 2 IndustryPack modules
	TCP220	Compact PCI carrier for 4 IndustryPack modules
carrier_sbs_pci	PCI40	PCI carrier for 4 IndustryPack modules
	cPCI100	Compact PCI carrier for 2 IndustryPack modules
	cPCI200	Compact PCI carrier for 4 IndustryPack modules
carrier_vme (SBS UniSDK)	TVME200	VMEbus carrier for 4 IndustryPack modules
	TVME201	VMEbus carrier for 4 IndustryPack modules
	TVME210	VMEbus carrier for 2 IndustryPack modules
	TVME211	VMEbus carrier for 2 IndustryPack modules
	TVME220	VMEbus carrier for 4 IndustryPack modules

5.2 Enumeration of IPAC slots

If more than one IPAC module is installed, maybe on different carrier boards. It's sometimes necessary to know which device node belongs to a certain slot on a carrier board.

The search and allocation order of the carrier class driver is always deterministic and never accidental. Usually the PCI bus will be searched from lower buses to higher buses and from lower devices to higher devices.

On carrier boards the slots will be enumerated from lower slots to higher slots.

If different carrier boards are installed in the system the order depends on the start order of the carrier port drivers. If the carrier port driver will be automatically started by the carrier class driver the start order depends on order of entries in the list *carrier_PnP_list* in the header file *./class/pnpinf.h*. If manually started, the order depends of course on the manually start order.

5.3 Exclude specific PCI Devices

To exclude some specific PCI devices, the exact location on the PCI bus can be specified in the structure *rejectedPciDevices* in the header file ipac_carrier.h. If a device is found matching the specified values, it is rejected by the carrier port driver.

The structure has the following layout:

```
struct PciDeviceStruct {
   unsigned char busNo;
   unsigned char devNo;
   unsigned char funcNo;
} PciDeviceStruct;
```

Members

busNo

This parameter specifies the PCI bus number, where the specific PCI device is mounted.

devNo

This parameter specifies the device number of the specific PCI device on the bus.

funcNo

This parameter specifies the function number of the specific PCI device.

To retrieve the necessary parameters, execute **`lspci`** or take a look into the file */proc/pci* and search for the desired device that should not be used by the carrier port driver.

```
# lspci
00:0f.0 Bridge: TEWS Datentechnik GmBH: Unknown device 3064
00:11.0 Bridge: TEWS Datentechnik GmBH: Unknown device 30c8
```

Example

```
/*
** This will exclude the following PCI devices located on bus 0:
** device 0x0f and device 0x11.
*/
#define MAX_REJECT_PCI_DEVICES     2
static PciDeviceStruct rejectedPciDevices[MAX_REJECT_PCI_DEVICES] = {
     {0x00, 0x0f, 0x00},
     {0x00, 0x11, 0x00} };
```

5.4 Diagnostic

If your installed IPAC port driver (e.g. tip903drv) doesn't find any devices although the IPAC is properly plugged on a carrier slot, it's interesting to now what's going on in the system.

5.4.1 /proc file system entry

The TEWS TECHNOLOGIES IPAC carrier driver exports detailed information of registered IP slots, of plugged IP modules and their configuration, of registered IP port drivers and low-level carrier drivers via the /proc file system. All these information can be retrieved by a simple cat to the /proc file system entry /proc/tews-ip-carrier. Most of the displayed information is of interest only to the device driver developer and should be added to a support request in case of trouble with the carrier driver respective IP port driver.

```
# cat /proc/tews-ip-carrier

TEWS TECHNOLOGIES - IPAC Carrier Class Driver version 1.3.0 (2006-06-07)

Registered IP slots:

[TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier - Slot 0]
    Plugged Module     Vendor=0xB3, Modul=0x1C
    Installed Driver   TIP903 - 3 Channel Extended CAN Bus IP -
    Slot Setup         INT0_EN | LEVEL_SENS | CLK_8MHZ | MEM_16BIT |
                       Memory Size = 0x400
    Interrupt Vector   System=5, Module=5
    Interrupt Level    INT0=0, INT1=0
    ID Space Addr      Physical=0xec821080, Virtual=0x26db5080
    IO Space Addr      Physical=0xec821000, Virtual=0x00000000
    MEM8 Space Addr    Physical=0xec000000, Virtual=0x00000000
    MEM16 Space Addr   Physical=0xeb000000, Virtual=0x26dca000

[TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier - Slot 1]
    Plugged Module     Vendor=0xB3, Modul=0x1D
    Installed Driver   TIP866 - 8 Channel Serial IP
    Slot Setup         INT0_EN | INT1_EN | LEVEL_SENS | CLK_8MHZ |
                       Memory Size = 0x0
    Interrupt Vector   System=5, Module=5
    Interrupt Level    INT0=0, INT1=0
    ID Space Addr      Physical=0xec821180, Virtual=0x26dcc180
    IO Space Addr      Physical=0xec821100, Virtual=0x26dff100
    MEM8 Space Addr    Physical=0xec400000, Virtual=0x00000000
    MEM16 Space Addr   Physical=0xeb800000, Virtual=0x00000000

Registered Carrier Drivers:
TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier V1.3.0
```

5.4.2 Debug Statements (printk())

Usually all TEWS TECHNOLOGIES device driver announced significant events or errors via the kernel message system (printk()).

You can retrieve this messages from the /proc file system using the following command

 # cat /proc/kmsg

```
TEWS TECHNOLOGIES - IPAC Carrier Class Driver version 1.3.0 (2006-06-07)

TEWS TECHNOLOGIES - Default Carrier version 1.3.0 (2006-06-07)

TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier version 1.3.0 (2006-06-07)

TIP903 - 3 Channel Extended CAN Bus IP - version 1.2.0 (2006-04-05)<6>

TIP903  :  Probe new TIP903 mounted on <TEWS TECHNOLOGIES - (Compact)PCI IPAC Carrier> at slot
A
```

If the standard and error messages doesn't help to locate the problem you can enable more detailed debug output in each driver by removing the comments around the DEBUGxxx definitions.

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.