The Embedded I/O Company

TEWS TECHNOLOGIES

CARRIER-SW-95

QNX-Neutrino Device Driver

IPAC Carrier Interface

Version 1.1.x

User Manual

Issue 1.1.0

October 2009

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

CARRIER-SW-95

QNX-Neutrino Device Driver

IPAC-Carrier Interface

Issue	Description	Date
1.0	First Issue	October 14, 2004
1.0.1	General Revision, new address TEWS LLC	August 26, 2008
1.1.0	Carrier Support for Tundra TSI148 VME added	October 12, 2009

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.

The IPAC and carrier devices drivers are implemented as libraries that will be linked to the IPAC class driver. Basically the concept implements a three layer model. The carrier port drivers are at the lowest layer, the middle layer will be handled by the IPAC class driver and the highest layer is 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 starting the IPAC class driver, the driver will recognize supported carrier boards by itself. It will start the required carrier port drivers; collect information about plugged IPAC modules and starts appropriate IPAC port drivers.

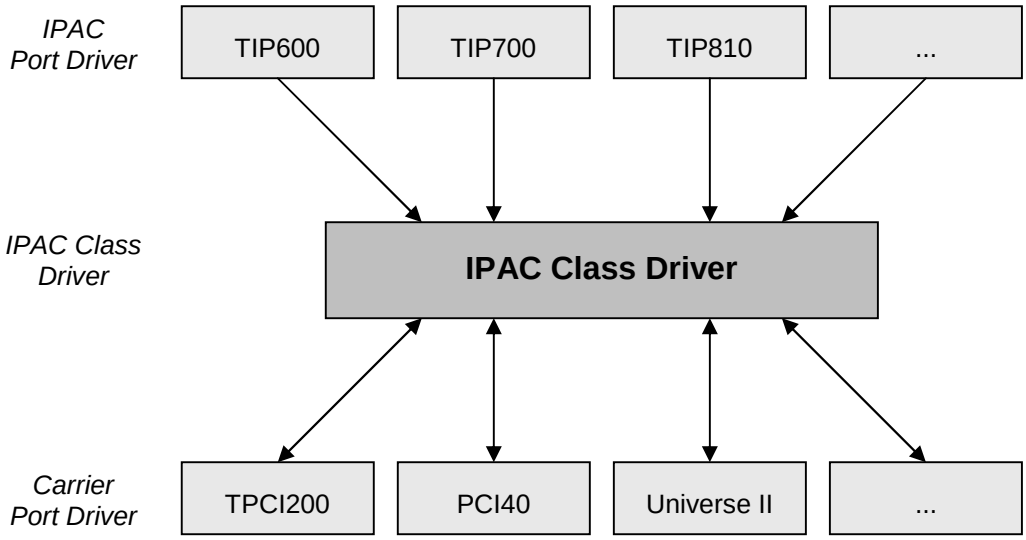

Figure 1: Driver Architecture

2 Installation

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

The directory CARRIER-SW-95 on the distribution media contains the following files:

CARRIER-SW-95-1.1.0.pdf	This manual in PDF format
CARRIER-SW-95-SRC.tar.gz	A tar archive containing the driver source code
ChangeLog.txt	Release history
Release.txt	Release information

The GZIP compressed archive CARRIER-SW-95-SRC.tar.gz contains the following files and directories:

Directory path 'ipac_carrier':

class	Subdirectory with IPAC class driver source code
carrier_tews_pci	Subdirectory with TEWS PCI carrier port driver source code
carrier_sbs_pci	Subdirectory with SBS PCI carrier port driver source code
carrier_universe_vme	Subdirectory with Universe/II VME carrier port driver source code and configuration file
carrier_tsi148_vme	Subdirectory with TSI148 VME carrier port driver source code and configuration file
include	Subdirectory with driver include files
ipac_carrier_driver.txt	carrier port driver list
ipac_driver.txt	IPAC port driver list

In order to perform an installation, copy CARRIER-SW-95-SRC.tar.gz to */usr/src* and extract all files of the archive. (`tar –xzf CARRIER-SW-95-SRC.tar.gz`).

After that the necessary directory structure for the automatic build and the source files are available beneath the new directory called ipac_carrier.

Additionally, copy the header files from directory "include" to /usr/include.

It is absolutely important to extract the archive in the /usr/src directory. Otherwise the automatic build with *make* will fail.

2.1 Build the IPAC class driver

Change to */usr/src/ipac_carrier/class* directory and execute the *Makefile*.

```
# cd /usr/src/ipac_carrier/class
# make install
```

After successful completion the driver will be installed in the */bin* directory

2.2 Build a carrier port driver

Change to the */usr/src/ipac_carrier/carrier_xxx_yyy* directory and execute the *Makefile*. The example will build the driver for TEWS PCI IPAC carrier boards.

```
# cd /usr/src/ipac_carrier/carrier_tews_pci
# make install
```

After successful compilation the driver library will be installed in */lib/dll* directory.

2.3 Modify carrier port driver list

Change to the */usr/src/ipac_carrier* directory. Open the file *ipac_carrier_driver.txt* with an editor. Allow the use of the carrier port driver with adding the library file name. Libraries not installed to the library default path */lib/dll* must be specified with the full path name.

All lines not empty or not starting with '#' will be interpreted as a path name.

After modifying this file, it must be copied to */etc/IPAC_CARRIER*.

2.4 Modify IPAC port driver list

Change to the */usr/src/ipac_carrier* directory. Open the file ipac_driver.*txt* with an editor. Add entries for all drivers and IPs that will be installed. All lines not empty or not starting with '#' will be interpreted as an entry.

An entry is split into four sections the first section specifies the name of the IPAC port driver library. Libraries not installed to the library default path */lib/dll* must be specified with the full path name. The second and third sections are selecting the module the driver should be used for. The second specifies the manufacturer ID (hex) and the third specifies the module ID (hex). The fourth section is just to comment the line. The sections are split by space or tab characters.

After modifying this file, it must be copied to */etc/IPAC_CARRIER*.

2.5 Start the IPAC class driver

The driver is started by calling *ipac_class*. Parameters can be used for debugging. Specifying '-v' enables verbose mode for the IPAC class driver and carrier port driver. Specifying '-V' enables verbose mode for the IPAC port driver.

Example starts with full debug information:

```
# ipac_class –v –V
```

The driver can be started in background mode with adding a '&' at the end of the command line.

Example starts without debug information in background:

```
# ipac_class &
```

3 Carrier Port Driver

3.1 TEWS PCI Carrier

This carrier port driver will setup the carriers and manage the accesses to the supported carriers.

3.1.1 Supported Carriers

The following carriers are supported by the TEWS PCI carrier port driver:

TPCI100	2 Slot PCI Carrier
TPCI200	4 Slot PCI Carrier
TCP201	4 Slot cPCI Carrier
TCP210	4 Slot cPCI Carrier
TCP211	2 Slot cPCI Carrier
TCP212	2 Slot cPCI Carrier
TCP213	2 Slot cPCI Carrier
TCP220	4 Slot cPCI Carrier

3.2 SBS PCI Carrier

This carrier port driver will setup the carriers and manage the accesses to the supported carriers.

3.2.1 Supported Carriers

The following carriers are supported by the TEWS PCI carrier port driver:

PCI40	4 Slot PCI Carrier
PCI60	6 Slot PCI Carrier
cPci100	2 Slot cPCI Carrier
cPci200	4 Slot cPCI Carrier

3.3 Universe VME Carrier

This carrier port driver will setup the Universe VME controller and handle the accesses to the VME bus.

This driver must not be used with other drivers setting up the VME controller.

3.3.1 Supported Carriers

The Universe VME carrier port driver supports the handling of VME carrier boards with a Tundra Universe based VME bus.

3.3.2 Configuration

Because the VME bus does not support Plug & Play, we have to use a configuration file to identify valid slots. This file must be adapted for the target system. The file with the VME configuration is named *uvmeConfig.txt* and a default is delivered with the driver. The configuration file must be copied into the */etc/IPAC_CARRIER* directory.

The configuration file is split into three sections, the global VME settings, VME master window settings, and IPAC slot settings. Each block starts with a keyword and is followed by a fixed number of parameters. Lines which are empty or starting with '#' are skipped when scanning the input.

To use a configuration matching to your system, you may have to adapt the values. After changing the file make sure it is copied into the */etc/IPAC_CARRIER* directory to use it.

3.3.2.1 Global VME Configuration

This block is started with the keyword **VMEBus* and followed by 9 lines of parameters. See the example with the delivered default values and the parameter value description below.

For detailed information please refer to the Tundra Universe/II documentation (Master Control Register: MAST_CTL, Miscellaneous Control Register: MISC_CTL).

Example (default values):

```
*VMEBus
00000000                # RequestMode
00000000                # ReleaseMode
00000001                # ArbitrationMode
00000010                # ArbitrationTimeout
00000040                # Timeout
00000003                # RequestLevel
00000008                # NumberOfRetries
00000200                # PostedWriteTransferCount
00000002                # SYSCON:
```

Parameters

All values are specified as hex values.

RequestMode

0	Demand (default)
1	Fair

ReleaseMode

0	Release When Done (RWD) *(default)*
1	Release on Request (ROR)

ArbitrationMode

0	Round Robin
1	Priority *(default)*

ArbitrationTimeout

0	Timeout 0µs
10	Timeout 16µs *(default)*
100	Timeout 256µs

Timeout

10	Timeout 16µs
20	Timeout 32µs
40	Timeout 64µs *(default)*
80	Timeout 128µs
100	Timeout 256µs
200	Timeout 512µs
400	Timeout 1024µs

RequestLevel

0	VMEBus Request Level 0
1	VMEBus Request Level 1
2	VMEBus Request Level 2
3	VMEBus Request Level 3 *(default)*

NumberOfRetries

0	Endless retries
n	*n* * 64 retries (*n* = 1..F) (*default* = 8)

PostedWriteTransferCount

80	128 byte
100	256 byte
200	512 byte *(default)*
400	1024 byte
800	2048 byte
1000	4096 byte

SYSCON

0	not system controller
1	system controller
2	automatic system controller selection *(default)*

3.3.2.2 VME Master Window Settings

This block is started with the keyword **Window n* and followed by 5 lines of parameters. The VME master window number is specified by *n*. Allowed VME master window numbers are 1..8. See the example with the delivered default values for VME master window 1 (A16/D16) and the parameter value description below.

Example (window 1 (A16/D16)):

```
*Window 1
00000001                # WindowEnable
00000000                # WindowStartAddress
00010000                # WindowSize
00000029                # AddressModifierCode
00000010                # DataWidth
```

Parameters

All values are specified as hex values.

WindowEnable

0	disable VME master window
1	enabled VME master window

WindowStartAddress

Specifies the start address of the VME master window.

WindowSize

Specifies the size of the VME master window in bytes.

AddressModifierCode

09	A32 non privileged data access
1A	A32 non-privileged program access
0D	A32 supervisory data access
0E	A32 supervisory program access
29	A16 non-privileged access
2D	A16 supervisory access
39	A24 non-privileged data access
3A	A24 non-privileged program access
3D	A24 supervisory data access
3E	A24 supervisory program access

DataWidth

08	8 bit data words
10	16 bit data words
20	32 bit data words

3.3.2.3 IPAC Slot Settings

This block is started with the keyword **Slot* and followed by 13 lines of parameters. See the example with the delivered default values for IPAC Slot 0 and the parameter value description below.

Example:

```
*Slot
00000001                # SlotEnable

00006080                # VmeIdAddress
00000080                # VmeIdSpaceSize
00000001                # VmeIdSpaceWindow

00006000                # VmeIoSpaceAddress
00000080                # VmeIoSpaceSize
00000001                # VmeIoSpaceWindow

00D00000                # VmeMemSpaceAddress
00040000                # VmeMemSpaceSize
00000002                # VmeMemSpaceWindow

000000A0                # BaseInterruptVector
00000001                # Int0VmeInterruptLevel
00000002                # Int1VmeInterruptLevel
```

Parameters

All values are specified as hex values. The parameters mainly depend on the used VME carrier board and its configuration, please have a look to the VME Carrier manual to find the matching values.

SlotEnable

0	disable IPAC slot
1	enabled IPAC slot

VmeIdSpaceAddress

Specifies the absolute VMEbus address where the ID space of the IPAC can be found.

VmeIdSpaceSize

Specifies the size of the ID space in bytes.

VmeIdSpaceWindow

Specifies the VME master window that is used for the ID space. Allowed values are 1..8.

VmeIoSpaceAddress

Specifies the absolute VMEbus address where the I/O space of the IPAC can be found.

VmeIoSpaceSize

Specifies the size of the I/O space in bytes.

VmeIoSpaceWindow

Specifies the VME master window that is used for the I/O space. Allowed values are 1..8.

VmeMemSpaceAddress

Specifies the absolute VMEbus address where the memory space of the IPAC can be found.

VmeMemSpaceSize

Specifies the size of the memory space in bytes.

VmeMemSpaceWindow

Specifies the VME master window that is used for the memory space. Allowed values are 1..8.

BaseInterruptVector

Defines a base interrupt vector used for the slot. 8 vectors will be reserved for the slot (BaseInterruptVector .. BaseInterruptVector+7). The vectors must be unique for the system. Valid values are 40h..F8h.

Int0VmeInterruptLevel

Specifies the VME interrupt level (1..7) for INT0.

Int1VmeInterruptLevel

Specifies the VME interrupt level (1..7) for INT1.

3.3.3 Inserting additional IPAC slots

For adding a new IPAC slot, simply copy one of the IPAC slot setting blocks and modify the parameters matching to the new slot.

After the modification of the configuration file is finished make sure the file is copied to the */etc/IPAC_CARRIER* directory to use these settings.

3.4 TSI148 VME Carrier

This carrier port driver will setup the TSI148 VME controller and handle the accesses to the VME bus.

This driver must not be used with other drivers setting up the VME controller.

3.4.1 Supported Carriers

The TSI148 VME carrier port driver supports the handling of VME carrier boards with a Tundra TSI148 based VME bus.

3.4.2 Preconditions to use this Carrier Driver

A PCI-memory space must be mapped on the PCI-PCI-bridge in front of the TSI148. This should be supported by the BIOS, the firmware or by the operating system during PCI-setup. This driver will place the TSI148 VME windows into this address space.

Some CPU boards support a more comfortable TSI148 configuration by BIOS or firmware. It may allow setting VME Windows and other VME settings. This driver allows the use of these configurations, but it also allows to overwrite them, or to extend them.

3.4.3 Configuration

Because the VME bus does not support Plug & Play, we have to use a configuration file to identify valid slots. This file must be adapted for the target system. The file with the VME configuration is named *tsi148vmeConfig.txt* and a default is delivered with the driver. The configuration file must be copied into the */etc/IPAC_CARRIER* directory.

The configuration file is split into three sections, the global VME settings, VME master window settings, and IPAC slot settings. Each section starts with a keyword followed by a fixed number of parameters. Lines which are empty or starting with '#' are skipped while scanning the input.

To use a configuration matching to your system, you may have to adapt the values. After changing the file make sure it is copied into the */etc/IPAC_CARRIER* directory to use it.

Please keep in mind that the complete configuration must match into the allocated PCI-space of the TSI148.

3.4.3.1 Global VME Configuration

This block is started with the keyword **VMEBus* and followed by 9 lines of parameters. See the example with the delivered default values and the parameter value description below.

For detailed information please refer to the Tundra TSI148 documentation (VME Master Control Register: LCSR_VMCTRL, VME Control Register: LCSR_VCTRL).

Example (default values):

```
*VMEBus
00000000          # VMEBus setting
00000000          # RequestMode
00000000          # RequestMode
00000000          # ReleaseMode
00000001          # ArbitrationMode
00000010          # ArbiterTimeout
00000040          # Timeout
00000003          # RequestLevel
00000040          # VME Master Time On
00000000          # VME Master Time Off
00000400          # VME Slave Deadlock Timer
```

Parameters

All values are specified as hex values.

VMEBusSetting

0	no VMEBus setting, use BIOS settings
1	enable VMEBus setting

RequestMode

0	Demand (default)
1	Fair

ReleaseMode

0	Release on TIME ON or DONE *(default)*
1	Release on (TIME ON and REQ) or DONE
2	Release on (TIME ON and BCLR) or DONE
3	Release on (TIME ON or DONE) and REQ

ArbitrationMode

0	Round Robin
1	Priority *(default)*

ArbiterTimeout

0	Arbiter Timeout Disabled
1	Arbiter Timeout Enabled

Timeout

0	Timeout disabled
8	Timeout 8µs
10	Timeout 16µs
20	Timeout 32µs
40	Timeout 64µs *(default)*
80	Timeout 128µs
100	Timeout 256µs
200	Timeout 512µs
400	Timeout 1024µs
800	Timeout 2048µs

RequestLevel

0	VMEBus Request Level 0
1	VMEBus Request Level 1
2	VMEBus Request Level 2
3	VMEBus Request Level 3 *(default)*

VME Master Time On

4	4µs (128 byte)
8	8µs (128 byte)
10	16µs (128 byte)
20	32µs (256 byte)
40	64µs (512 byte) *(default)*
80	128µs (1024 byte)
100	256µs (2048 byte)
200	512µs (4096 byte)

VME Master Time Off

0	0µs *(default)*
1	1µs
2	2µs
4	4µs
8	8µs
10	16µs
20	32µs
40	64µs

VME Slave Deadlock Timer

0	Deadlock Retry Disabled
10	16 VCLKs
20	32 VCLKs
40	64 VCLKs
80	128 VCLKs
100	256 VCLKs
200	512 VCLKs
400	1024 VCLKs *(default)*
800	2048 VCLKs
1000	4096 VCLKs
2000	8192 VCLKs
4000	16384 VCLKs
8000	32768 VCLKs

3.4.3.2 VME Master Window Settings

This block is started with the keyword **Window n* and followed by 5 lines of parameters. The VME master window number is specified by *n*. Allowed VME master window numbers are 1...8. See the example with the delivered default values for VME master window 1 (A16/D16) and the parameter value description below.

Example (window 1 (A16/D16)):

```
*Window 1
00000001                # WindowEnable
00000000                # WindowStartAddress
00010000                # WindowSize
00000029                # AddressModifierCode
00000010                # DataWidth
00000000                # PrefetchSize
```

Parameters

All values are specified as hex values.

WindowEnable

0	disable VME master window
1	enabled VME master window
2	skip VME master window setup, use BIOS configuration

WindowStartAddress

Specifies the start address of the VME master window.

WindowSize

Specifies the size of the VME master window in bytes.

AddressModifierCode

09	A32 non privileged data access
1A	A32 non-privileged program access
0D	A32 supervisory data access
0E	A32 supervisory program access
29	A16 non-privileged access
2D	A16 supervisory access
39	A24 non-privileged data access
3A	A24 non-privileged program access
3D	A24 supervisory data access
3E	A24 supervisory program access

DataWidth

10	16 bit data words
20	32 bit data words

PrefetchSize

0	Memory Read Prefetch Disabled
2	2 Cache Lines
4	4 Cache Lines
8	8 Cache Lines
10	16 Cache Lines

3.4.3.3 IPAC Slot Settings

This block is started with the keyword **Slot* and followed by 13 lines of parameters. See the example with the delivered default values for IPAC Slot 0 and the parameter value description below.

Example:

```
*Slot
00000001                # SlotEnable

00006080                # VmeIdAddress
00000080                # VmeIdSpaceSize
00000001                # VmeIdSpaceWindow

00006000                # VmeIoSpaceAddress
00000080                # VmeIoSpaceSize
00000001                # VmeIoSpaceWindow

00D00000                # VmeMemSpaceAddress
00040000                # VmeMemSpaceSize
00000002                # VmeMemSpaceWindow

000000A0                # BaseInterruptVector
00000001                # Int0VmeInterruptLevel
00000002                # Int1VmeInterruptLevel
```

Parameters

All values are specified as hex values. The parameters mainly depend on the used VME carrier board and its configuration; please have a look to the VME Carrier manual to find the matching values.

SlotEnable

0	disable IPAC slot
1	enabled IPAC slot

VmeIdSpaceAddress

Specifies the absolute VMEbus address where the ID space of the IPAC can be found.

VmeIdSpaceSize

Specifies the size of the ID space in bytes.

VmeIdSpaceWindow

Specifies the VME master window that is used for the ID space. Allowed values are 1...8.

VmeIoSpaceAddress

Specifies the absolute VMEbus address where the I/O space of the IPAC can be found.

VmeIoSpaceSize

Specifies the size of the I/O space in bytes.

VmeIoSpaceWindow

Specifies the VME master window that is used for the I/O space. Allowed values are 1...8.

VmeMemSpaceAddress

Specifies the absolute VMEbus address where the memory space of the IPAC can be found.

VmeMemSpaceSize

Specifies the size of the memory space in bytes.

VmeMemSpaceWindow

Specifies the VME master window that is used for the memory space. Allowed values are 1...8.

BaseInterruptVector

Defines a base interrupt vector used for the slot. 8 vectors will be reserved for the slot (BaseInterruptVector .. BaseInterruptVector+7). The vectors must be unique for the system. Valid values are 40h..F8h.

Int0VmeInterruptLevel

Specifies the VME interrupt level (1...7) for INT0.

Int1VmeInterruptLevel

Specifies the VME interrupt level (1...7) for INT1.

3.4.4 Inserting additional IPAC slots

For adding a new IPAC slot, simply copy one of the IPAC slot setting blocks and modify the parameters matching to the new slot.

After the modification of the configuration file is finished make sure the file is copied to the */etc/IPAC_CARRIER* directory to use these settings.

4 Appendix

4.1 Enumeration of IPAC slots

The IPAC slots are used in the same direction as they are recognized on the bus or in the configuration file. This direction depends on the list of the carrier port drivers specified in the carrier port driver list and is inverted. Next point is the PCI search direction of the QNX-Neutrino. And at last the enumeration on a carrier is always from starting at slot A up to the last slot or the enumeration in the configuration. All empty slots will not be counted.

4.2 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.

4.3 IPAC Port Driver Support

All used IPAC port drivers must match to the IPAC carrier driver concept of TEWS TECHNOLOGIES.

If your IPAC isn't supported for this IPAC carrier driver please contact TEWS TECHNOLOGIES.