

Release Note



VxWorks Driver Installation for NetBlaster PCI and CompactPCI Fast Ethernet Adapters

Installing VxWorks 5.3.x (Intel Platforms) Driver

These instructions are relevant to the VxWorks driver for ZNYX NetBlaster PCI and CompactPCI adapters, models ZX345Q, ZX346Q, ZX348Q, ZX412 and ZX414. The VxWorks driver for NetBlaster adapters can be obtained from either the ZNYX web site at <http://www.znyx.com> or ftp site at <ftp.znyx.com>. The distribution is a self-extracting archive containing all the files needed to complete installation.

RETRIEVING DRIVERS

1. Visit the Driver Download area of the ZNYX web site to obtain the appropriate VxWorks driver for your NetBlaster adapter:
<http://www.znyx.com/drivers/drivers.htm>
2. The driver is a self-extracting archive. Save the file and make note of its location.
3. Execute the file to extract the driver and information files. The README.TXT file contains the instructions presented below.

INSTALLING DRIVERS

1. Copy the driver files `if_znb.obj`, `if_zxePci.c` and `if_zxePci.h` into the `$(WIND_BASE)/target/config/$(BSP)` directory

NOTE: This is not a SENS driver and will not link in a Tornado environment with SENS installed.

2. Modify the "Makefile" in the `$(WIND_BASE)/target/config/$(BSP)` directory by changing the line:

```
MACH_EXTRA =  
  
to  
  
MACH_EXTRA = if_znb.obj
```

If there is already a value assigned to MACH_EXTRA, simply add "if_znb.obj" to the list.

3. Modify the file `$(WIND_BASE)/target/config/$(BSP)/config.h` to add the following definitions...

```
/* Network driver options */  
.  
.  
.
```

```

#define INCLUDE_ZXE          /* include ZNYX support */
#ifndef INCLUDE_PCI         /* include PCI bus library */
#define INCLUDE_PCI
#endif
#ifndef STANDALONE_NET     /* for ZNYX bootable floppy */
#define STANDALONE_NET
#endif
#ifndef LOGIN_USER_NAME
#define LOGIN_USER_NAME    "target" /* for testing */
#define LOGIN_PASSWORD    "bReb99RRed" /* "password" */
#endif

```

4. Modify the file \$(WIND_BASE)/target/config/\$(BSP)/pc.h to add the following definitions:

```

/* FEI PCI bus resources */
.
.
.
/* Znyx PCI bus resources */

#define ZXE0_MEMBASE0    0xed000000
#define ZXE0_IOBASE0    0xF800
#define ZXE0_MEMSIZE0    0x1000
#define ZXE0_INT_LVL    0xb
#define ZXE0_INT_VEC    (INT_NUM_IRQ0 + ZXE0_INT_LVL)
#define ZXE1_MEMBASE0    0xed100000
#define ZXE1_IOBASE0    0xF880
#define ZXE1_MEMSIZE0    ZXE0_MEMSIZE0
#define ZXE1_INT_LVL    ZXE0_INT_LVL
#define ZXE1_INT_VEC    ZXE0_INT_VEC
#define ZXE2_MEMBASE0    0xed200000
#define ZXE2_IOBASE0    0xFC00
#define ZXE2_MEMSIZE0    ZXE0_MEMSIZE0
#define ZXE2_INT_LVL    ZXE0_INT_LVL
#define ZXE1_INT_VEC    ZXE0_INT_VEC
#define ZXE3_MEMBASE0    0xed300000
#define ZXE3_IOBASE0    0xFC80
#define ZXE3_MEMSIZE0    ZXE0_MEMSIZE0
#define ZXE3_INT_LVL    ZXE0_INT_LVL
#define ZXE3_INT_VEC    ZXE0_INT_VEC
#define ZXE4_MEMBASE0    0xed400000
#define ZXE4_IOBASE0    0xFD00
#define ZXE4_MEMSIZE0    ZXE0_MEMSIZE0
#define ZXE4_INT_LVL    ZXE0_INT_LVL
#define ZXE4_INT_VEC    ZXE0_INT_VEC
#define ZXE5_MEMBASE0    0xed500000
#define ZXE5_IOBASE0    0xFD80
#define ZXE5_MEMSIZE0    ZXE0_MEMSIZE0

```

```

#define ZXE5_INT_LVL      ZXE0_INT_LVL
#define ZXE5_INT_VEC      ZXE0_INT_VEC
#define ZXE6_MEMBASE0    0xed600000
#define ZXE6_IOBASE0     0xFE00
#define ZXE6_MEMSIZE0    ZXE0_MEMSIZE0
#define ZXE6_INT_LVL     ZXE0_INT_LVL
#define ZXE6_INT_VEC     ZXE0_INT_VEC
#define ZXE7_MEMBASE0    0xed700000
#define ZXE7_IOBASE0     0xFE80
#define ZXE7_MEMSIZE0    ZXE0_MEMSIZE0
#define ZXE7_INT_LVL     ZXE0_INT_LVL
#define ZXE7_INT_VEC     ZXE0_INT_VEC

#define ZXE_INIT_STATE_MASK (VM_STATE_MASK_VALID | \
                             VM_STATE_MASK_WRITABLE | \
                             VM_STATE_MASK_CACHEABLE)
#define ZXE_INIT_STATE (VM_STATE_VALID | \
                        VM_STATE_WRITABLE | \
                        VM_STATE_CACHEABLE_NOT)

```

5. Modify the file \$(WIND_BASE)/target/config/\$(BSP)/sysLib.c to add the following change to the sysPhysMemDesc[] structure making sure these new entries come before any DUMMY_MMU_ENTRY if present:

```

PHYS_MEM_DESC sysPhysMemDesc [] =
{
.
.
.

#ifdef INCLUDE_ZXE
{
(void *) ZXE0_MEMBASE0,
(void *) ZXE0_MEMBASE0,
ZXE0_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
{
(void *) ZXE1_MEMBASE0,
(void *) ZXE1_MEMBASE0,
ZXE1_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
{
(void *) ZXE2_MEMBASE0,
(void *) ZXE2_MEMBASE0,
ZXE2_MEMSIZE0,
ZXE_INIT_STATE_MASK,

```

```

ZXE_INIT_STATE
},
{
(void *) ZXE3_MEMBASE0,
(void *) ZXE3_MEMBASE0,
ZXE3_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
{
(void *) ZXE4_MEMBASE0,
(void *) ZXE4_MEMBASE0,
ZXE4_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
{
(void *) ZXE5_MEMBASE0,
(void *) ZXE5_MEMBASE0,
ZXE5_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
{
(void *) ZXE6_MEMBASE0,
(void *) ZXE6_MEMBASE0,
ZXE6_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
{
(void *) ZXE7_MEMBASE0,
(void *) ZXE7_MEMBASE0,
ZXE7_MEMSIZE0,
ZXE_INIT_STATE_MASK,
ZXE_INIT_STATE
},
#endif
.
.
.
};

```

Add the following include:

```

.
.
.
#include "sysNetif.c"          /* network driver support */

```

```

#ifdef INCLUDE_ZXE
#include "if_zxePci.c"
#endif
.
.
.

```

Add the following to the initialization:

```

.
.
.
sys557PciInit ();
#endif /* INCLUDE_FEI */

#ifdef INCLUDE_ZXE
zxePciInit ();
#endif /* INCLUDE_ZXE */
.
.
.

```

4. Move to the \$(WIND_BASE)/target/config/\$(BSP) directory and compile a new standalone kernel by entering

```
make vxWorks.st
```

5. Make a copy of the bootable floppy from Wind River and save the original. The new kernel image vxWorks.st can be copied to the new bootable floppy.
6. Issue the following commands from the console after vxWorks has booted:

```

-> znbattach <unit>,0,0,0,0,<mediaType>
-> ifMaskSet "znb<unit>",0xFFFFFFFF
-> ifAddrSet "znb<unit>","XX.XX.XX.XX"

```

SUBROUTINES

znbattach()

NAME *znbattach()* - publishes the **znb** network interface and initialize the driver and device

SYNOPSIS

```
int znbattach (  
  int unit, /* device unit, 0-7 */  
  unsigned long devAdrs, /* I/O base address, must be 0 */  
  int ivec, /* interrupt vector, must be 0 */  
  int ilevel, /* interrupt level, must be 0 */  
  unsigned long pciMemBase, /* main memory base, must be 0 */  
  int mediaType /* media type, 0-7 */  
)
```

DESCRIPTION This routine publishes the **znb** interface by filling in a network interface record and adding that record to the system list. It also initializes the driver and the device to the operational state.

The following media types are defined:

MEDIA_AUTO	0	/* auto negotiation */
MEDIA_TP	1	/* 10 Mbit - half duplex */
MEDIA_BNC	2	/* not supported */
MEDIA_AUI	3	/* not supported */
MEDIA_TPFD	4	/* 10 Mbit - full duplex */
MEDIA_TX	5	/* 100 Mbit - half duplex */
MEDIA_TXFD	6	/* 100 Mbit - full duplex */
MEDIA_T4	7	/* not supported */

RETURNS OK or ERROR

znbVersion()

NAME *znbVersion()* - returns the version of the znb driver

SYNOPSIS

```
int znbVersion ( )
```

DESCRIPTION Returns the version of the znb driver

RETURNS OK or ERROR



ZNYX Corporation
48501 Warm Springs Blvd., Suite 107
Fremont, CA 94539 USA
(510) 249-0800
Fax (510) 656-2460
www.znyx.com

Document # DU0128-01

© 1998 ZNYX Corporation. All rights reserved worldwide. All information in this document is subject to change without prior notice. ZNYX and NetBlaster are trademarks or registered trademarks of ZNYX Corporation in the United States and/or other countries. All other marks, trademarks or service marks are the property of their respective owners.