Intel® IXP400 Software Release 1.4

Software Release Notes

April 20, 2004

Contents

Tables

1.0 Important Information About This Software Release

This document provides important information for installing and using the Intel® IXP400 Software v.1.4 release, which is intended to be used with the *Intel® IXP400 Software Programmer's Guide* (Document Number: 252539-005).

2.0 What's New

The following major additions have been made to software release 1.4. For more specific information, see the *Intel® IXP400 Software Programmer's Guide*.

- The Linux Ethernet Driver and Kernel Integration patches have been modified to allow upgrades of the Ethernet Driver to occur independently from the MontaVista* Linux kernel integration patches. The change to those patches is now reflected in these release notes.
- The software is now supported on MontaVista Linux Professional Edition 3.0 and 3.1, Wind River Systems* VxWorks* 5.5.1/Tornado* 2.2.1, and Wind River PLATFORM for Network Equipment* 2.0
- Several access-layer components have been enhanced or modified. These generally include:
  - IxEthAcc performance enhancements and support for larger Ethernet frames. Additionally, a change was made that requires the *IX_MBUF_NEXT_PKT_IN_CHAIN_PTR* mBuf field to be a NULL value, or it will trigger the IX_ASSERT condition and fail. IxEthAcc now uses this field internally and returns this field to NULL before passing it back up the stack. In previous releases, this field was not used by IxEthAcc.

    Important: **Care must be taken when porting code that used a previous release of this component, since the new version checks for a NULL value in this field.**
  - IxCryptoAcc now supports ARC 4 encryption/decryption with WEP ICV authentication for Wired Equivalent Privacy (WEP) applications, and AES-CCM mode cryptographic and authentication processing for 802.11i applications.
  - Support for 56K Raw and HDLC channel has been added to IxHssAcc. The `IxHssAccPktPortConnect()` function has been modified to accommodate these new features and **is not backwards compatible** with the same function in previous releases.
  - The IxPerfProfAcc component has been enhanced two ways: it displays a larger number of tasks and it integrates symbols, which weren't previously integrated.
  - Little-endian support has been extended to the ADSL Driver, IxPerfProfAcc, and several codelets.
  - The IxFpathAcc component and codelet have been de-featured.
- The Linux Ethernet driver performance has been improved by taking advantage of Intel XScale core preload instructions, improving driver initialization, and better integrating with the kernel-queueing mechanisms.
- The software base directory now defaults to `/ixp400_xscale_sw`.
- The API reference documentation is now provided with the software release, instead of appearing as an appendix in the *Intel® IXP400 Software Programmer's Guide*.

3.0 Additional Resources

The following table lists additional documents relevant to this release. These documents are available from your field representative or from:

www.intel.com/design/network/products/npfamily/docs/ixp4xx.htm.

Title	Document Number
Intel® IXP400 Software Programmer's Guide	252539
Intel® IXP400 Software Specification Update	273795
Intel® IXP400 Software: Red Hat Boot-Loader v.1.92 Software - Software Release Notes*	N/A†
Intel® IXP4XX Product Line and IXC1100 Control Plane Processors Datasheet	252479
Intel® IXP4XX Product Line of Network Processors and IXC1100 Control Plane Processors Specification Update	252702
Intel® IXP4XX Product Line and IXC1100 Control Plane Processors Developer's Manual	252480

† This document is on the software Web page at:
http://www.intel.com/design/network/products/npfamily/ixp425swr1.htm

4.0 Obtaining the Intel® IXP400 Software

These release notes walk the user through the steps of obtaining the necessary software, preparing the host and development systems, building a target-based image that includes the IXP400 software, and utilizing one of the supplied codelets to demonstrate software's functionality.

To obtain the IXP400 software:

1. Go to the download link at the following Web site:

   http://www.intel.com/design/network/products/npfamily/ixp425swr1.htm

   All software provided by Intel is available by using the "Download" link.

2. Acquire the version of the Intel® IXP400 Software for the desired operating system.

   Each version has two versions of the software release:

   - The complete version *that contains the cyptographic components* is named **ixp400AccessLibraryWithCrypto-1_4.zip**.
   - The version that *contains all components except for cryptography* is called **ixp400AccessLibrary-1_4.zip**.

3. *If using the Linux operating system*, obtain the "Associated Intel Linux Device Driver and Integration patches for MVL PE 3.0/3.1."

   While on this site, be sure to check for any updated drivers or patches that may apply to your situation.

   As a convenience, the RedBoot* boot-loader source code and tool chain are available from the Intel Web site listed above.

4. Proceed to the instructions specific to the operating system being used.

   - For *VxWorks*, see "Installing and Building the Software — VxWorks* and PLATFORM for Network Equipment 2.0" on page 7.
   - For *Linux* development, see "Installing and Building the Software — Linux" on page 19.

5.0 Installing and Building the Software — VxWorks* and PLATFORM for Network Equipment 2.0

At a minimum, VxWorks 5.5.1 / Tornado 2.2.1 is required with Intel® IXP400 Software v.1.4.

The Wind River PLATFORM for Network Equipment 2.0 (PNE) is also supported and provides an enhanced IP stack. PNE is also based on VxWorks 5.5.1/Tornado 2.2.1, therefore installation instructions for both environments are substantially similar.

This section documents both environments.

5.1 Little-Endian Support

Intel® IXP400 Software v.1.4 supports operation in little-endian mode with VxWorks 5.5.1/ Tornado 2.2.1 and PLATFORM for Network Equipment 2.0 (PNE).

Little-endian support requires the following modifications to the build process, which are also documented through these release notes:

- A B-0 stepping of the IXP4XX product line or IXC1100 control plane processor.
- A little-endian version of the VxWorks boot loader.
  See "Building Boot ROM Images" on page 14.
- A little-endian compiled version of PNE.
  See Step 2 of "Building a VxWorks* Image with the Access Library" on page 9
- A little-endian compiled version VxWorks image, by changing the IX_TARGET definition.
  See Step 4 of "Building a VxWorks* Image with the Access Library" on page 9.

5.2 Installing Tornado*, the VxWorks* BSP, and PNE

Prerequisite: The Intel® IXDP425 / IXCDP1100 Development Platform ships with a VxWorks 5.5 big-endian boot ROM. If the IXDP425 / IXCDP1100 platform contains a VxWorks 5.4 boot ROM, it must be upgraded.

Refer to "Using ixFlashUpgrade to Upgrade a Tornado* 2.1.1 Boot ROM" on page 17.

1. Install Tornado 2.2.1 and the "BSP/Drivers for VxWorks 5.5.1" from the CD that comes with the Intel® IXDP425 / IXCDP1100 Development Platform.
   - If you are using the *Tornado 2.2.1 Evaluation CD* provided with the Intel® IXDP425 / IXCDP1100 Development Platform, an installation key is required.

     For information on obtaining an installation key, go to the following Web site:

     http://www.windriver.com/partnerships/eval-cd

- On the board-support package (BSP)/Drivers CD, the only required fields for the "Select Products" window are:
  - BSP Developers Kit
  - Driver Objects/Headers: intel-arm
  - Network Driver Source Code

  Note: Do **NOT** use the IXDP425 BSP that is provided on this CD. It is out-of-date.

2. Obtain and install the desired, KIXDP425BD-specific BSP.

   The **Version 1.2/8 BSP** for the IXDP425 / IXCDP1100 platform can be obtained from the Wind River Systems Web site given in Step 1.

   The IXDP425 / IXCDP1100 platform **BSP** is available in the **Downloads** section of that site, or by searching for "ixdp425."

3. Apply the SPR 89608 BSP patch, located under the BSP code directory. For example:

   `c:\tornado\target\config\ixdp425\patch-SPR89608`.

   This patch enables P-bit support in the processor MMU to enable little-endian support.

4. *If using PNE*, install the PLATFORM NE software according to the instructions provided by Wind River Systems.

5.3 Installing the Access Library

1. Extract the contents of the file `ixp400AccessLibrary-1_4.zip` (or `ixp400AccessLibraryWithCrypto-1_4.zip`) into an empty directory.

   This file contains the Intel® IXP400 Software v.1.4 access library and demo codelets source code. Typically, this file is extracted to the default `tornado` directory and results in a subdirectory called `ixp400_xscale_sw` being created. For example:

   c:\tornado\ixp400_xscale_sw

2. Change the values of the WIND_BASE, CSR_BASE, and IX_TARGET environmental variables by editing one of the following files in the new directory:

   - For Windows*:
     `ixp400_xscale_sw\buildUtils\environment.vxworks.bat`
   - For Bourne-shell-compatible Unix shells:
     `ixp400_xscale_sw/buildUtils/environment.vxworks.sh`
   - For C-Shell-compatible Unix shells:
     `ixp400_xscale_sw/buildUtils/environment.vxworks.csh`

   a. Modify the WIND_BASE variable to point to the Tornado 2.2.1 installation directory. For example: `WIND_BASE=c:\tornado`

   b. Modify the CSR_BASE environment variable to point to where the IXP400 software is installed.
   For example: `CSR_BASE=c:\tornado\ixp400_xscale_sw`.

   c. Modify the value for the variable IX_TARGET.
   - The IX_TARGET variable defaults to `vxbe`, or "VxWorks Big-Endian."
   - To build for a little-endian target, change this value to `vxle`.

   For more information on IX_TARGET, see "Makefile Macros" on page 29.

3. Set up the environment by executing one of the following commands:
   - For Windows command shell:
     `buildUtils\environment.vxworks.bat`
   - For Bourne-shell compatible Unix shells:
     `. buildUtils/environment.vxworks.sh`
   - For C-Shell compatible Unix shells:
     `source buildUtils/environment.vxworks.csh`

5.4 Building a VxWorks* Image with the Access Library

This section demonstrates how to build a loadable vxWork.st image that will include the IXP400 software access library and BSP code.

1. Ensure the environment is set — per Step 3 of "Installing the Access Library" on page 8 — before giving the make command.
2. *If using PNE*, you can enable the IP fast-path router stack for improved performance. To use this feature:
   a. Change to the directory:
      `c:\tornado\target\src\wrn\make`
   b. Enter the following command:

```
make CPU=XSCALE TOOL=gnube ADDED_CFLAGS+=-DROUTER_STACK ADDED_CFLAGS+=-O3
```

NOTES:
1. If using a little-endian target, replace `TOOL=gnube` with `TOOL=gnu`.
2. The entire command must be on a single line.

   c. Depending on the endianness of the target processor, edit the file `c:\tornado\target\config\ixdp425\config.h` or `c:\tornado\target\config\ixdp425_le` to add the following line in the execution path of the file:

```
#define INCLUDE_FASTPATH
```

NOTE: Line 550 is the recommended location for this code.

3. Change the working directory to where the `ixp400AccessLibrary-1_4.zip` file was extracted.
   For example: `c:\tornado\ixp400_xscale_sw`
4. Build the IXP400 software access library by entering the following command:

```
make libIxp425
```

NOTE: libIxp425 is case-sensitive with capital "I."

The resulting file libIxp425.a is placed in:

`c:\tornado\ixp400_xscale_sw\lib\<IX_TARGET>`

Note: The IXP400 software and the associated VxWorks BSP support both big- and little-endian targets. The makefile supports these different targets by specifying an IX_TARGET macro value. VxWorks / Big-Endian is the default target type, having the value of `vxbe`.

In the preceding example, `libIxp425.a` would be placed in `c:\tornado\ixp400_xscale_sw\lib\vxbe`.

For more information, see "Makefile Macros" on page 29.

5. Build a VxWorks image by entering the command:

```
make vxWorks.st
```

NOTE: VxWorks is case-sensitive with capital "W."

There now should be a VxWorks image in the BSP directory with filename `vxWorks.st`. There are two BSP directories, one for each endian version of the target. For example: `c:\tornado\target\config\ixdp425` or `c:\tornado\target\config\ixdp425_le`.

5.5 Loading a VxWorks* Image with the Access Library

This section demonstrates how to:

- Configure the boot ROM on the IXDP425 / IXCDP1100 platform to communicate with a Tornado* host machine
- Download a loadable vxWork.st image that will include the IXP400 software access library and BSP code

1. Physically connect the IXDP425 / IXCDP1100 platform to the host machine running Tornado, according to the *Intel® IXDP425 / IXCDP1100 Development Platform Quick Start Guide*.
2. The VxWorks boot ROM will need to download the newly created vxWorks.st image.
   a. On the host system running Tornado, start an FTP server service (such as WSFTP).
   b. Set up a user/password pair for the VxWorks boot ROM, and ensure that the FTP service provides access to the vxWorks.st image created in "Building a VxWorks* Image with the Access Library" on page 9.
   In this example, the vxWorks.st file is located at:
   `c:\tornado\target\config\ixdp425`.

To configure the WFTPD application included with Tornado 2.2.1:

1. Start WFTPD by selecting **Start** > **Programs** > **Tornado 2.2.1** > **FTP Server**.
2. Create a new user and password.
   In this example, the following values are used:
   - User: ***target***
   - Password: ***target***
3. Set the home directory for the FTP service.
   In this example, the following path is used:
   `C:\`

4. Open a terminal emulator with a connection to the UART 1 port of the IXDP425 / IXCDP1100 platform.

   Tornado creates default HyperTerminal settings through the command series **Start** > **Programs** > **Tornado 2.2.1** > **VxWorks COM1**.

   Serial port settings for VxWorks are ***9600-8-N-1***.

5. Configure the Tornado host PC network settings.

   This example uses the following network settings:

   - IP: `192.168.0.100`
   - Subnet Mask: `255.255.255.0`
   - Gateway: `not required`

6. Boot the board.

7. When prompted, press a key to stop the boot sequence.

   Following the successful initialization of the boot ROM, the seven-segment LED should read `0027`, and the terminal output should read as shown in Table 1.

Table 1. Console Output of Successful VxWorks* Boot ROM Load

```
 VxWorks System Boot



Copyright 1984-2002  Wind River Systems, Inc.




CPU: Intel IXP425 - IXDP425 BE
Version: VxWorks5.5
BSP version: 1.3/3
Creation date: Jun 11 2003, 15:26:30



Press any key to stop auto-boot...
 6
[VxWorks Boot]:
```

8. Configure the boot ROM parameters to connect to the Tornado host machine and download the image.

   - To view current settings, press **p**.
   - To change settings, press **c**.

   Note: For an explanation of the boot-ROM parameters, refer to Table 2 through Table 4.

Table 2. Displaying the VxWorks* Boot ROM parameters

```
[VxWorks Boot]: p

boot device          : fei
unit number          : 0
processor number     : 0
host name            : ixdp425
file name            :
c:\Tornado\target\config\ixdp425\vxWorks.st
inet on ethernet (e) : 192.168.0.50
host inet (h)        : 192.168.0.100
gateway inet (g)     :
user (u)             : target
ftp password (pw)    : target
flags (f)            : 0x0
other (o)            : ixe
```

Table 3. VxWorks* Boot-ROM Parameter Values (Sheet 1 of 2)

Parameter	Values	Description
boot device	fei0, ixe0, ixe1	The Ethernet device being configured to connect to the Tornado* host. • fei0 — The Intel 8255x-based PCI card • ixe0 — The integrated Ethernet port (“Ethernet 0 Module” in center of the board) • ixe1 — The integrated Ethernet port (“Ethernet 1 Module” closest to the UART ports)
file name	(Varies)	The relative path from FTP root to the VxWorks file. If the root directory of the FTP service is `c:\Tornado\target\config` The value here would be `\ixdp425\vxWorks.st`.
inet on ethernet (e)	xxx.xxx.xxx.xxx or xxx.xxx.xxx.xxx:FFFFFF00 where FFFFFF00 is the subnet mask in hex.	IP address being assigned to the boot device. The subnet mask may also be supplied, if required.
host inet (h)	xxx.xxx.xxx.xxx or xxx.xxx.xxx.xxx:FFFFFF00 where FFFFFF00 is the subnet mask in hex.	IP address of the Tornado host. The subnet mask may also be supplied, if required.
user (u)	(Varies)	FTP service user name

Table 3. VxWorks* Boot-ROM Parameter Values (Sheet 2 of 2)

Parameter	Values	Description
password (pw)	(Varies)	FTP service password for user.
flags (f)	• 0x00 — Use FTP to get boot image • 0x02 — Load local system symbols • 0x04 — Don't autoboot • 0x08 — Quick autoboot (no countdown) • 0x20 — Disable login security • 0x40 — Use bootp to get boot parameters • 0x80 — Use tftp to get boot image • 0x100 — Use proxy arp	VxWorks Boot ROM parameters.
other (o)	ixe, or <blank>	Entering `ixe` instructs the BSP to load the Ethernet END driver, to enable VxWorks to use the onboard ixe0 and ixe1 Ethernet devices. **NOTE:** If using the IxEthAccCodelet, see Section 5.8.4.

9. When the VxWorks boot-ROM parameters have been set, press **@** to boot the board.

   A VxWorks shell will appear in the terminal window. Additionally, the VxWorks Ethernet END driver will display the current configuration of the onboard IXE0 and IXE1 Ethernet interfaces.

Table 4. VxWorks* Shell After Successful Boot of Loadable VxWorks* Image

```
ixdp425EthEndStartUp: ixe0: IP Addr set to 192.168.50.1
ixdp425EthEndStartUp: ixe1: IP Addr set to 192.168.60.1
->
```

10. Set up Tornado 2.2.1 to communicate with the IXDP425 / IXCDP1100 platform by using the following procedure:
    a. Start Tornado 2.2.1 by selecting **Start** > **Programs** > **Tornado 2.2.1** > **Tornado**.
    b. Close the Create Project dialogue box.
    c. Select **Tools** > **Target Server** > **Configure**.
    d. Click the **New** button.
    e. In the **Available Back Ends** field, select ***wdbrpc***.
    f. In the **Target Name / IP Address** field, type the IP address of the development board's boot device.

       In this example, it is `192.168.0.50`.
    g. Press the **Launch** button.

       An icon will appear in the task bar.
    h. In the drop-down box in the Tornado 2.2.1 GUI, select the newly created target and click the **Launch Shell** button (the icon resembles **`->i`**).

       A new Windshell* window should appear in the Tornado GUI.

    The VxWorks image now is running on the IXDP425 / IXCDP1100 platform.

At this point, the following actions are available to the developer:

- Use of the software release 1.4 access library functions
  Refer to the *Intel® IXP400 Software Programmer's Guide*.
- Building and running of loadable modules, such as the codelets
  Refer to "Building Demo Codelets" on page 15.
- Configuration of the IXE Ethernet interfaces, using the `ixdp425IfConfig` function
  Refer to "Enabling the IXE Ethernet Interface in VxWorks*" on page 16.
- Enabling of the PNE Fast-Path IP stack
  Refer to "Enabling PNE IP Fast-Path" on page 16

5.6 Building Boot ROM Images

1. Ensure the environment is set before giving the make command.
   See Step 3 of "Installing the Access Library" on page 8.
2. Change the working directory to where the `ixp400AccessLibrary-1_4.zip` file was extracted. For example:
   `c:\tornado\ixp400_xscale_sw`
3. By default, the boot-ROM requires the IXP400 software access library to be created first. To do this, type the following:

`make clean`	To remove any objects from previous builds
`make libIxp425`	To create the IXP400 Access Library

4. Change the working directory to where the IXDP425 BSP is installed. For example:
   `c:\tornado\target\config\ixdp425` —or—
   `c:\tornado\target\config\ixdp425_le`
5. Open `config.h`. Go to line 557 of `config.h` and verify that it equals `#if 0`. If not, change it.
6. Type one of the following:
   - To remove any objects from previous builds:

```
make clean
make bootrom.bin
```

   - Otherwise:

```
make bootrom.hex
```

The BSP directory now should contain a file called `bootrom.bin` or `bootrom.hex`. This is a boot-ROM image in binary or hex format.

At this point, any flash burner — visionClick* or the IxFlashUpgrade utility — can be used to program the file into the flash.

- For instructions on using visionClick, refer to the BSP documentation, mentioned in "VxWorks* BSP and END-Driver Documentation" on page 18.

- For instructions on the IxFlashUpgrade utility, see "Using ixFlashUpgrade to Upgrade a Tornado* 2.1.1 Boot ROM" on page 17.

5.7 Building Demo Codelets

1. Ensure the environment is set before giving the make command.
   See Step 3 of "Installing the Access Library" on page 8.
2. Change the working directory to where the `ixp425AccessLibrary-1_3.zip` file was extracted.
3. Do one of the following to get a codelet on the IXDP425 / IXCDP1100 platform:

   Note: For a list of available codelets, refer to "Makefile Macros" on page 29.

   - If the board is currently booted to a working VxWorks image, use a loadable module by:
     a. Build the ethAcc codelet by using the following command:

     ```
     make loadable COMP=codelets_ethAcc
     ```

     The loadable object file is placed in:
     `c:\tornado\ixp400_xscale_sw\lib\<IX_TARGET>`

     b. To download the loadable module, ensure the board is booted with a valid VxWorks image and — in a Tornado Windshell window — enter the command:

```
ld < c:\tornado\ixp400_xscale_sw\lib\vxbe\codelets_ethAcc.out ()
```

NOTE: Use the specific directory names for the host machine.

   - To create a VxWorks image that includes a codelet:
     a. Build a vxWorks image with a codelet by using the following command:

     ```
     make vxWorks.st COMP=codelets_ethAcc
     ```

     The resulting file VxWorks is placed in:
     `c:\tornado\target\config\ixdp425` (or `ixdp425_le`)

     b. Load the new VxWorks image file according to "Loading a VxWorks* Image with the Access Library" on page 10.
4. To demonstrate the codelet listed above, enter the following command in the Tornado Windshell* window or in the terminal editor:

```
ixEthAccCodeletMain()
```

5.8 IXE Ethernet Interface Support in VxWorks*

5.8.1 Booting from the IXE Ethernet Interface in the VxWorks* Boot ROM

The IXDP425 / IXCDP1100 platform can be enabled to boot to a remote VxWorks image from either of the embedded IXE Ethernet interfaces.

In order to do this, change the *boot device* field in the boot ROM configuration menu to the value of ***ixe0*** or ***ixe1***. This procedure is described in "Loading a VxWorks* Image with the Access Library" on page 10.

5.8.2 Enabling the IXE Ethernet Interface in VxWorks*

The VxWorks image includes support for the embedded IXE Ethernet interfaces through the use of the IXE END driver. By default, this driver is loaded in VxWorks.

In order to *enable* the loading of the IXE END driver, change the *other* field in the boot ROM configuration menu to the value of ***ixe***. To *disable*, change the *other* field to any value other than ***ixe***. (This procedure is described in "Loading a VxWorks* Image with the Access Library" on page 10.)

Note: If the IXE interface is used as a boot device by the boot ROM, the driver will also be enabled in the VxWorks OS image, regardless of the value used in the *other* field of the boot ROM configuration. If the user wishes to use the IXE as a boot device, but disable it in the OS, the driver should be excluded when building the OS image as described in "Including or Excluding the IXE END Driver" on page 16.

5.8.3 Including or Excluding the IXE END Driver

The IXE END Driver can be included or excluded when building the VxWorks image or boot ROM by doing to following:

1. Edit the file `config.h` in the BSP directory.
   For example:
   `c:\tornado\target\config\ixdp425\config.h`
2. Locate the line `#define INCLUDE_IXETHACCEND`. To exclude the IXE END Driver, change the line to `#undef INCLUDE_IXETHACCEND`.
3. Re-build the VxWorks image or boot ROM, as shown in "Building a VxWorks* Image with the Access Library" on page 9.

5.8.4 ixEthAccCodelet and the IXE END Driver

The ixEthAccCodelet will not run if the VxWorks IXE END driver is running. The IxEthAccCodelet uses additional Intel® IXP400 Software access-layer components in order to configure and send/receive traffic with the embedded Ethernet MAC devices. The VxWorks IXE END driver will conflict with these components if used simultaneously.

In order to use the ixEthAccCodelet on VxWorks, disable the VxWorks IXE END driver by following the instructions in "Enabling the IXE Ethernet Interface in VxWorks*" on page 16.

5.8.5 Enabling PNE IP Fast-Path

The PLATFORM for Network Equipment IP fast-path feature must be configured to enable the feature on both of the NPE-based Ethernet interfaces. This feature only works on routed traffic between the IXE 0 and IXE 1 interfaces.

For more information on this feature, refer to the Wind River WIND* NET Router Stack documentation.

To enable IP fast-path, enter the following command in a Tornado Windshell or terminal shell on the target board, as documented in "Loading a VxWorks* Image with the Access Library" on page 10:

```
-> ffInit
-> ffIntEnable "ixe", 0
-> ffIntEnable "ixe", 1
-> ffModeSet 1
```

5.9 Using ixFlashUpgrade to Upgrade a Tornado* 2.1.1 Boot ROM

Intel® IXP400 Software v.1.2.1 and later releases require a minimum of Tornado 2.2 (VxWorks Version 5.5). The software will not work on a development board that has a Tornado 2.1.1 (VxWorks Version 5.4.2) boot ROM in flash.

The flash-upgrade utility included in this release may be used to upgrade such a board to a Tornado 2.2.1 boot ROM.

Note: The ixFlashUpgrade utility can only be used to program big-endian boot-ROM images.

The following instructions assume that the board is currently running with Tornado 2.1.1 and that a Tornado 2.1.1-compatible release of this software (for example, Intel® IXP400 Software v.1.1) is accessible.

1. On a Tornado 2.2.1 (or higher) host system with the current IXP400 software installed, build a Tornado 2.2.1 boot ROM image.
   - In the Tornado 2.2.1 directory, navigate under `target/config/ixdp425`
   - Typing the following:

     ```
     make bootrom.bin
     ```

2. Place the file `bootrom.bin` on a TFTP server accessible from the development board.
3. Copy the current release's source code for the flashUpgrade component into a new directory called `src/flashUpgrade`, of the `ixp425_xscale_sw` directory on a Tornado 2.1.1-based host system with a compatible IXP400 software installation (for example, Intel® IXP400 Software v.1.1).
4. On the Tornado 2.1.1-based host system, edit the file `ixp425_xscale_sw/Makefile` to add the word `flashUpgrade` to the list of components on the line beginning with `COMPONENTS :=`.
5. Build the flash-upgrade utility so that it can be executed on the Tornado 2.1.1 board by doing one of the following:
   - To build a standalone image, enter the following:

     ```
     make libIxp425
     make vxWorks.st COMP=flashUpgrade
     ```

OR...

- To build a loadable module in the `ixp425_xscale_sw` directory:

```
make libIxp425
make loadable COMP=flashUpgrade
```

When the build is complete the file `flashUpgrade.out` is in the directory `ixp425_xscale_sw\lib\armobjs`.

6. Reboot the board by doing one of the following:
   - Boot with the new `vxWorks.st` image containing flashUpgrade

     OR...
   - Boot an existing `vxWorks.st image` and load the flashUpgrade module separately. For example: By running `ld < flashUpgradeTest.out` in a WindShell.
7. Execute the following command to download the Tornado 2.2.1 boot-ROM image via TFTP and burn it into flash:

```
ixFlashUpgrade "-l", "<tftp server address>", "<bootrom location>"
```

Example: The TFTP server IP address is `17.17.17.121` and the boot ROM image is located on that server, at `tftpboot/user/bootrom.bin`. The command would be:

```
ixFlashUpgrade "-l", "17.17.17.121", "user/bootrom.bin"
```

A Tornado 2.2.1 boot ROM is now burned into the flash.

8. Reboot.

   The VxWorks version number `5.5.1` should appear in the console output during boot-up, indicating that the board has been upgraded successfully.

5.10 VxWorks* BSP and END-Driver Documentation

For specific information on the IXDP425 / IXCDP1100 platform BSP, see the Tornado 2.2.1 online manuals. These are available within the Tornado GUI via the **Help > Manuals Index** menu option, and then selecting the ***VxWorks BSP Reference*** book.

Alternately, it may be accessed directly at the following path:

`c:\tornado\docs\vxworks\bsp\ixdp425\ixdp425.html`
(or `ixdp425_le.html` for little-endian targets).

6.0 Installing and Building the Software — Linux

6.1 Upgrading from a Previous Release

There are two basic steps:

- Archive the existing access library and kernel source tree
- Start with a fresh copy of the Linux Support Package (LSP) kernel source tree

  Do not start with a kernel source tree which has had patches from a previous release applied to it.

6.2 Directory Variables

Several variables are used in this document to specify the location of specific MontaVista Linux Professional Edition or IXP400 software components. There are some differences in these directory locations depending on the version of MontaVista Linux being used. This table lists the placeholder variables or names used in this document and the default locations used by the supported versions of MontaVista Linux.

Table 5. Default Directory Values

Variable	MVL Version	Default Value
<path-to-lsp>	3.0	`/opt/hardhat/devkit/lsp/intel-ixdp425-arm_xscale_be`
	3.1	`/opt/montavista/pro/devkit/lsp/intel-ixdp4xx-arm_xscale_be`
HARDHAT_BASE	3.0	`/opt/hardhat`
	3.1	`/opt/montavista/pro`
LSP Kernel Directory	3.0	`/opt/hardhat/devkit/lsp/intel-ixdp425-arm_xscale_be/linux-2.4.18_mvl30`
	3.1	`/opt/montavista/pro/devkit/lsp/intel-ixdp4xx-arm_xscale_be/linux-2.4.20_mvl31`

6.3 Prerequisites

Before starting these installation instructions, do the following:

- Install the MontaVista Linux Professional Edition 3.0 or 3.1 (MVL PE 3.0 / 3.1) General Availability release
- Install the IXDP425 / IXCDP1100 platform's LSP

  This step includes setting up the development host system.

For MontaVista Linux 3.0 — These steps are documented in the *MontaVista Linux Professional Edition for the Intel® IXDP425 / IXCDP1100 Development Platform Quick Start Guide*. The document is located at the following site:

http://support.mvista.com/content/Pro/Documentation30/quickstarts/intel-ixdp425-pe-qs.pdf

Note: Access to the document listed above requires support from MontaVista.

For MontaVista Linux 3.1 — These steps are documented in the *MontaVista® Linux® Professional Edition 3.1 Cross-Development Quick Start Guide* HTML document located at `\docs\quickstart\cross-development\index.html` on the Host CD.

6.4 Installation

1. Make a working directory by typing the following command:

   ```
   mkdir <workdir>
   ```

   For the purposes of this procedure the working directory is annotated as `<workdir>`

2. If not already done as part of the MontaVista Linux setup procedure, set up the LSP kernel source tree in the working directory. To set up the LSP kernel manually, copying the LSP kernel into a directory under <workdir>.

   a. For MVL 3.0:

   ```
   $ cp -a <path-to-lsp>/linux-2.4.18_mvl30 <workdir>/
   ```

   b. For MVL 3.1:

   ```
   $ cp -a <path-to-lsp>/linux-2.4.20_mvl31 <workdir>/
   ```

   This places the LSP kernel in a directory under `<workdir>` as `linux-2.4.xx_mvl3x`.

For convenience, the directory can be renamed `linux`. Alternatively, create a soft link named `linux` that points to the directory, so the LSP kernel version is maintained in the directory name.

Note: From this point forward in the instructions, the LSP kernel source tree is accessed as if it is under the directory `<workdir>/linux`.

3. Extract the contents of the access library zip file into the working directory. Execute by typing one of the following:
   - If using the *standard access* library:

```
$ cd <workdir>
$ unzip <path-to-file>/ixp400AccessLibrary-1_4.zip
```

   - If using the *crypto-enabled* access library:

```
$ cd <workdir>
$ unzip <path-to-file>/ixp400AccessLibraryWithCrypto-1_4.zip
```

   Both files contain the Intel® IXP425 Network Processor access library and demo codelet source code. In either case, after following the commands, the access library source code is unzipped into the following directory:

   `<workdir>/ixp400_xscale_sw`

   At this point the working directory tree will have the following structure:

```
<workdir>
   |--ixp400_xscale_sw
   \--linux (or linux-2.4.xx_mvl3x)
```

4. Configure the access library build environment by editing one of the following files, changing the values of `LINUX_SRC`, `HARDHAT_BASE`, and `IX_XSCALE_SW` to reflect the directory layout for the host machine:
   - For Bourne-compatible shells

```
$ ixp400_xscale_sw/buildUtils/environment.linux.sh
```

   - For C-Shell compatible shells

```
$ ixp400_xscale_sw/buildUtils/environment.linux.csh
```

`LINUX_SRC`	Should point to `<workdir>/linux`
`HARDHAT_BASE`	Should point to the root of your MontaVista installation: `/opt/hardhat` or `/opt/montavista/pro`
`IX_XSCALE_SW`	Should point to the ixp400_xscale_sw directory: `<workdir>/ixp400_xscale_sw`

5. Set up the environment using one of the following commands:

```
$ source ./ixp400_xscale_sw/buildUtils/environment.linux.sh
```

   Or

```
$ source ixp400_xscale_sw/buildUtils/environment.linux.csh
```

   Note: The script sets the environment variables that allow the access library to be compiled with the Linux kernel. Each time the terminal in which the environment settings were originally set is exited, the script must be re-run in order for the environment settings to be available.

6. Obtain the most recently available Intel IXP400 Linux Ethernet Driver and Integration patch zip files.

   The Intel IXP400 Software Releases Web site — mentioned in "Obtaining the Intel® IXP400 Software" on page 6 — will direct you to the appropriate patch zip file(s). These files may be different than those listed below.

7. Extract the zip files into the working directory.

```
$ unzip <path-to-file>/ixp400LinuxIntegrationPatch-<version>.zip
$ unzip <path-to-file>/ixp400LinuxEthernetDriverPatch-<version>.zip
```

8. Apply the contained patch files while in the `<workdir>/linux` directory. If applicable, refer to any `readme.txt` files that were provided on the Web site with the zip file. Unless instructed differently in a `readme.txt`, the general process for applying patches is as follows:

```
$ cd linux
$ patch -p1 < ../ixp400LinuxKernel-<version>.patch
$ patch -p1 < ../ixp400LinuxEthernetDriver-<version>.patch
```

   Note: In general, the IXP400 Linux patches are used to update the kernel source to integrate the NPE-enabled Ethernet driver and associated kernel `.config` file and makefile changes. Source for the IXP400 Linux Ethernet driver is also provided along with these patches.

6.5 Preparing the Host Development System

A number of steps now must be taken to support cross-platform development between the Linux host-development system and the target IXDP425 / IXCDP1100 platform. This procedure is documented in the quick start guides, referenced in "Prerequisites" on page 19.

The general cross-development setup referred to in the release notes requires that these services are set up and configured correctly:

- A DHCP server running on the host machine, to provide an IP address for the RedBoot* boot loader on the development platform and the Linux kernel.
- A TFTP service running on the host machine to allow the RedBoot boot loader to download the kernel image.
- An NFS service running on the host machine for supporting the default kernel configuration that mounts the remote file system for the target.
- An exported NFS directory on the host machine — that provides a remote file system for the development platform — and a place from which to install kernel modules.
- A serial connection using the UART 0 port on the development board, and a terminal emulator (such as Minicom*) on the host platform set to 115200 baud, 8-N-1.
- The instructions in this release notes document assume that the host system and the development board are connected via a cross-over Ethernet cable using the Intel 8255x-based PCI card in the development board.

6.6 Building a Bootable Kernel Image

The commands are executed in the top-level Linux directory. As specified in the preceding section, that directory is designated as `<workdir>/linux`.

1. Make sure the environment is set.
   (See step 5 in "Installation" on page 20.)
2. Configure the kernel tree by running the following:

```
$ make ixdp425_config
$ make oldconfig
```

[In MVL 3.1, you will be prompted to select specific kernel options. Press **[Enter]** to accept the default values for all prompts **except for CONFIG_MTD_COMPLEX_MAPPINGS**. Select **Y** for the CONFIG_MTD_COMPLEX_MAPPINGS option.

For additional clarification of kernel configuration options, contact MontaVista.]

```
$ make dep
```

3. Build the Linux kernel image, execute the command:

```
$ make zImage
```

   This will create a file named `arch/arm/boot/zImage`, that can be booted using the RedBoot boot loader.
4. If necessary, copy the `zImage` file to the tftpboot directory, typically `/tftpboot`.

For instructions on loading the image, see "Using RedBoot* V1.92 to Load MontaVista* Linux Kernel Images" on page 25.

6.7 Building the Kernel Modules

The access library and codelets are built as kernel modules. They can be built by either of the two methods listed in this section.

6.7.1 Building the Kernel Modules as Part of the Kernel Build

This method builds all available kernel modules and installs them in a location accessible by the target development platform.

To build the access library and codelets:

1. Change the working directory to the Linux kernel source directory (`<workdir>/linux`).
2. Make sure the environment is set.
   (See step 5 in "Installation" on page 20.)

3. Build the configured kernel modules, including the IXP400 software access library (`ixp400.o`), NPE-ethernet driver (`ixp425_eth.o`) and demo codelets (`ixp400_codelets_xxx.o`), using the command:

```
$ make modules
```

4. Place the loadable modules into the embedded target file system on the host machine by using one of the following methods:
   - If the target file system is in the *standard* location or setup via the Target Configuration Tool (TCT):

```
$ make modules_install
```

   - If the target file system is *not* in the standard location:

```
$ make modules_install INSTALL_MOD_PATH=/path...
```

The value `/path...` is the path to the root of the target file system. This command copies the modules to the target file system in the correct directory structure.

- For MVL PE 3.0 the default target path is:
  `/opt/hardhat/devkit/arm/xscale_be/target`
- For MVL PE 3.1 the default target path is:
  `/opt/montavista/pro/devkit/arm/xscale_be/target`

6.7.2 Individually Building the Intel® IXP400 Software Kernel Modules

If the kernel is already configured properly and the developer wishes to modify and/or build the IXP400 access library by itself, this procedure can be followed. This assumes that the kernel has already been built and is properly configured.

1. Ensure that the environment is set.
   (See Step 5 in "Installation" on page 20.)
2. Change the working directory to the access library source directory:
   `<workdir>/ixp400_xscale_sw`
3. To build a module containing the access library components, use the command:

```
$ make ixp400.o
```

4. To build a module containing one of the codelets, use a command such as:

```
$ make module COMP=codelets_ethAcc
```

The resulting modules are located in the directory `ixp400_xscale_sw/lib/linuxbe/`.

6.8 Building the NPE-Ethernet Driver

The IXDP425 / IXCDP1100 platform provides two NPE-based Ethernet devices, in addition to a Intel 8255x-based PCI card. The NPE Ethernet devices are supported by a driver (`ixp425_eth.o`) that is implemented as a standard Linux network driver.

It is placed in the kernel source tree under the `drivers/net` directory when the `ixp400LinuxEthernetDriver-<version>.zip` update is applied to the kernel source.

The command `make modules` compiles this driver during the kernel build.

6.9 Loading a Kernel Image and the Intel® IXP400 Software with RedBoot*

6.9.1 Using RedBoot* V1.92 to Load MontaVista* Linux Kernel Images

This procedure assumes that the IXDP425 / IXCDP1100 platform has RedBoot pre-programmed into flash. For more information on how to program the flash images, refer to the Intel® IXP400 Software document *Red Hat Boot Loader Software v.1.92 Release Notes*.

1. It is recommended that the MAC addresses for the development board be set at this point. This process is documented in the Redboot 1.92 release notes, Section 6.2.
2. Load the kernel using the RedBoot commands:

```
Redboot> fis init -f
Redboot> load -v -r -b 0x06000000 zImage
Redboot> exec 0x06000000
```

Note: When updating the flash for the first time after loading an RedBoot image, run the command `fconfig -init` to initialize the RedBoot configuration.

6.9.2 Installing and Running the NPE-Ethernet Driver

After the zImage above is booted on the development board, follow the procedure shown below in order to load the kernel modules for the access-layer and NPE-Ethernet driver.

```
192.168.0.100 login: root

root@192.168.0.100:~# cd /
root@192.168.0.100:~# cd lib/modules/2.4.xx_mvl3x-ixdp425/kernel/drivers/ixp400
root@192.168.0.100:~# insmod ixp400.o
root@192.168.0.100:~# cd ../net
root@192.168.0.100:~# insmod ixp425_eth.o
```

At this point, the loadable modules containing the Intel® IXP400 Software and the Ethernet driver for the NPE-based interfaces are loaded.

6.9.3 Installing and Running Intel® IXP400 Software Codelet Modules

The IXP400 software contains example codelets that demonstrate the functionality of various access-layer components.

Some codelets (`IxEthAccCodelet`, for example) may not function if the embedded Ethernet driver (`ixp425_eth.o`) is loaded since the codelet accesses the Ethernet hardware on the NPE directly. Additionally, only one codelet may be loaded at any one time.

The commands listed below will load the IxEthAcc codelet and execute a loop-back test between IXP 0 and IXP 1. To test this codelet, connect an Ethernet cross-over cable between IXP 0 and IXP 1, then follow the instructions below.

If the embedded Ethernet driver is already loaded (from "Installing and Running the NPE-Ethernet Driver" on page 25), the board must be re-initialized first.

```
192.168.0.100 login: root

root@192.168.0.100:~# cd /
root@192.168.0.100:~# cd lib/modules/2.4.xx_mvl3x-ixdp425/kernel/drivers/ixp400
root@192.168.0.100:~# insmod ixp400.o
oot@192.168.0.100:~# insmod ixp400_codelets_ethAcc.o operationType=3
```

6.10 Additional RedBoot* Information

6.10.1 RedBoot*

RedBoot is the boot loader that is used to boot Linux. The binaries for RedBoot Release 1.92 can be obtained from:

http://www.intel.com/design/network/products/npfamily/ixp425swr1.htm

The quickest way to take advantage of NPE-enabled RedBoot is to use the binaries from Intel. These binaries are certified by Red Hat* and allow the NPE Ethernet interfaces to be immediately used on the standard platforms.

6.10.2 Automatic 'Load-and-Go' of the Linux* Kernel

RedBoot is capable of loading the kernel automatically at boot. This is accomplished by a simple script that is setup using the `fconfig` command.

To set up for automatic “load and go,” use the following commands:

```
Redboot> load -v -r -b 0x06000000 zImage
Redboot> fis create kernel -b 0x06000000 -l 0x000b0000
Redboot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis load kernel
>> exec 0x06000000
>>
Boot script timeout( 10mc resolution): 300
```

[*When additional choices are presented, press the* `Enter` *key to use the default value*]

```
Redboot> reset
```

NOTE: User-entered values are shown in **bold** type for clarity.

RedBoot reboots the development board and reloads. The following message is displayed:

```
== Executing boot script in 3.000 seconds - enter ^C to abort.
```

After a three-second pause, RedBoot executes the boot script to “load and go.” The kernel decompresses, boots, initializes, and NFS mounts its root file system.

Note: The length (size) of the kernel image (`-l 0x000b0000`) used in `fis create` command may be different depending on the specific configuration of the kernel. Adjust this value using the size identified after the image is initially loaded into RAM.

7.0 Intel® IXP400 Software Makefile

7.1 Makefile Commands

The following is a list of additional makefile options supported by software release 1.4. These commands are only applicable when using the top-level makefile located in the following directory:

`/ixp400_xscale_sw` directory.

Table 6. General Makefile Commands

Command	Description, Notes
make clean	Deletes all files that can be rebuilt by the build system. This includes object files, library files and dependency information. **NOTE:** A `make clean` should be performed any time a MACRO value is changed in order to ensure the new MACRO values take effect.

Table 7. VxWorks*-Only Make Commands

Command	Description, Notes
`make vxWorks COMP=(component)`	Builds a bootable VxWorks operating system image, suitable for download to a board, by invoking the BSP makefile. The BSP makefile will link in all of the production code objects for the components listed in <APP_COMPS>, along with any objects from libIxp425.a referenced by the specified component. These components are typically codelets.
`make vxWorks.st COMP=(component)`	Same as the previous command, this builds a standalone image that contains the specified component. These components are typically codelets.
`make loadable COMP=(component)`	This command builds a relocatable .out image of component code and component dependencies. These components are typically codelets.
`make libIxp425`	Builds a single library file containing all of the Intel XScale core library components.

Table 8. Linux*-Only Make Commands:

Command	Description, Notes
`make module COMP=(component)`	Builds a Linux kernel module of component code.
`make modules`	Builds all of the Linux kernel modules
`make ixp400.o`	Builds a Linux kernel module containing the access drivers

7.2 Makefile Macros

Parameters are passed to the build system as make macro definitions, either by defining them as environment variables or as command line arguments to make in the following form:

```
<macro_name>=<value>
```

Warning: The settings of any macros other than IX_TARGET and COMP only affect code rebuilt during the current `make` invocation. A `make clean` should be done when any values are changed to ensure that the changes propagate to the code structure.

Table 9. Makefile Macros

Macro	Values	Description
COMP	A software component name or codelet name prefixed by "`codelet_`".	Identifies the software component when building loadable object files of codelets or access layer components.
IX_TARGET	"vxbe" "vxle" "vxsim" "linuxbe"	Controls whether code is built for Intel XScale core hardware (vxbe (big-endian), vxle(little-endian)), the Solaris* VxWorks simulator (vxsim) or Linux platform (linuxbe).
IX_CFLAGS IX_LDFLAGS	Any legal GCC compiler flags	Additional compiler or linker flags to use when compiling code.
IX_DEBUG	Any value, or undefined	Enables additional parameter checks and trace messages. If set to any value, the C macro IX_DEBUG will be defined for all source code. Otherwise, the C macro NDEBUG will be defined.
IX_NOSYM	Any value, or undefined	If set to any value, no debugging symbol information is included in the build products.
IX_NOOPT	Any value, or undefined	Disable compiler optimizations
IX_CONTROLLED_COUNTRY_BUILD	Any value, or undefined	Builds an image without cryptography support. Cryptography features must not be released to "controlled countries," as defined by US export laws
BSP	The name of a BSP directory under Tornado's target/config directory, e.g. "ixdp425".	Specifies the BSP makefile to invoke when building a VxWorks OS image, and BSP-specific header files to include.

7.3 Available Codelets

This table lists the codelets available in this release. These values can be used with the COMP macro.

codelets_atm
codelets_cryptoAcc
codelets_dmaAcc†
codelets_ethAal5App
codelets_ethAcc
codelets_hssAcc
codelets_perfProfAcc
codelets_timers
codelets_usb

† Supported in big-endian mode only.

8.0 Access Library Source Code Documentation

The Intel® IXP400 Software compressed file contains two versions of source-code documentation:

- HTML-based source code documentation at:
  `ixp400_xscale_sw\doc\index.html`
- Printable form — in the Adobe* Acrobat* Portable Document Format (PDF) — at
  `ixp400_xscale_sw\doc\APIReference.pdf`

The access library source code uses a commenting style that supports the Doxygen* source code documentation system. Doxygen is an open-source tool, that reads appropriately commented source code and produces hyper-linked documentation of the APIs suitable for online browsing (HTML).

The documentation output is typically multiple HTML files, but Doxygen can be configured to produce LaTeX*, RTF* (Rich Text Format), PostScript*, hyper-linked PDF, compressed HTML and Unix* man pages. Doxygen is available for Linux, Windows* and other operating systems.

For more information, see:

http://www.doxygen.org.

9.0 Issues with Third-Party Components

This section provides the details reported as issues when using third-party software and tools. It is provided as a convenience for the developer. These are not considered errata in the Intel® IXP400 Software.

For a detailed list of software release 1.4 errata, see the *Intel® IXP400 Software Specification Update*.

Note: There may exist additional errata with third-party software and tools. Refer to the third-party vendor for additional information.

0556 Wind River Systems* visionICE* May Fail While Programming the Intel® IXDP425 / IXCDP1100 Development Platform's Flash

Issue: When attempting to program the Intel StrataFlash® device (E2F8128J) with the Wind River Systems visionICE* tool, the progress indicator may stop at about 50% and return the error message `ERROR #17: TF ALGORITHM TIMED OUT`.

Workaround: If `ERROR#17: TF ALAGORITHM TIMED OUT` occurs a "Retry" needs to be issued to successfully complete the programming of the flash device.

1024 Wind River Systems* VxWorks* 5.5 BSP for the Intel® IXDP425 / IXCDP1100 Development Platform Restricts the Outbound AHB-PCI Address Translation, by Default, to a Single, 16-Mbyte Window

Issue: The Wind River Systems VxWorks BSPs for the IXDP425 / IXCDP1100 platform defaults to supporting a single, 16-Mbyte AHB-PCI address translation window. In each BSP there is a file named `ixp425Pci.h` that defines PCIMEMBASE as 0x0. This define is written to the pci_pcimembase register which causes the four 16-Mbyte AHB-PCI address translation windows to point to the same PCI address range. When the pci_pcimembase register is set to 00 00 00 00, it causes `Membase0=00`, `Membase1=00`, `Membase2=00`, and `Membase3=00`. The PCI controller uses the `MembaseX` to replace the upper eight bits of the AHB address to translate it to the PCI address.

Implication: Only a single, 16-Mbyte PCI address space is support and the PCI address spaces for any target PCI devices hosted by the platform must fit within this address space. Also, different AHB addresses will map to the same PCI addresses. For example:

AHB 0x4800_1000 corresponds to PCI 0x0000_1000. Similarly, AHB 0x4B00_0000 will map to the same PCI address 0x0000_1000 since the four base address fields in pci_pcimembase register are 00 00 00 00.

Since the Wind River Systems BSP is provided in source code, a developer can modify this define for their system.

2439 Higher CPU Utilization for PCI Transactions with MontaVista* Linux 3.X

Issue: Intel XScale core utilization may be significantly higher for PCI transactions in MontaVista Linux 3.x when compared to MontaVista Linux 2.1. MontaVista Linux 3.x defaults to utilizing 256 Mbyte of SDRAM, while MontaVista 2.1 defaults to utilizing 64 Mbyte. Since the IXP4XX product line and IXC1100 control plane processors support a maximum memory window of 64 Mbyte, a configuration using 256 Mbyte cannot map PCI buffers directly to PCI address space.

The current MontaVista Linux 3.x code creates a safe buffer in the lower, 64-Mbyte address space and copies all data between the real buffers and this safe buffer in order to support a 256-Mbyte configuration. Configurations using 64 Mbyte of SDRAM with MontaVista Linux 3.x will have lower core utilization for PCI transactions, as long as the kernel is re-configured to use 64 Mbyte of SDRAM as opposed to the default 256 Mbyte.

This can be done by setting `mem=64M@0x00000000` on the kernel command line.

Implication: Core utilization will be higher when using a system configured with 256 Mbyte of SDRAM.

2665 MontaVista* Linux LSP Defines Reserved Expansion Bus Configuration Registers

Issue: In the file <LINUX_SRC>/include/asm/arch/ixp425.h, the following registers are defined:

```
#define IXP425_EXP_CFG0_OFFSET 0x20
#define IXP425_EXP_CFG1_OFFSET 0x24
#define IXP425_EXP_CFG2_OFFSET 0x28
#define IXP425_EXP_CFG3_OFFSET 0x2C
```

The expansion bus registers EXP_CFG2 and EXP_CFG3 are actually reserved and should not be modified.

Implication: Instead of four expansion bus configuration registers, the processor only provides two expansion bus configuration registers.

2808 Make VxWorks* Fails When Using Wind River Systems* VxWorks* BSP 1.2/7 with Tornado* 2.2 / VxWorks* 5.5

Issue: When attempting to build the VxWorks image using the BSP version 1.2/7 on a Tornado 2.2-based system, the build will fail. This is due to fact that the BSP supports some items that are only available in Tornado 2.2.1 / VxWorks 5.5.1.

Workaround: BSP 1.2/7 will work with Tornado 2.2 if the support for RFC 2233 is disabled in the BSP config.h file. To remove RFC 2233, perform the following steps:

1. Edit the file WIND_BASE/target/config/ixdp425/config.h or the WIND_BASE/target/config/ixdp425_le/config.h (depending on whether a big-endian or little-endian target is being used).
2. On line 511, change the entry #define INCLUDE_RFC_2233 to #undef INCLUDE_RFC_2233.
3. Save the config.h file.

VxWorks will now build properly.

2887 Incorrect Tools in Path Cause 'Make' to Fail with Wind River Systems* Tornado* Development Environment

Issue: If the development host system has duplicate copies of certain tools in the host system's path, an incorrect tool version could be called by the makefile during the build process. This is more likely to occur on host systems with multiple versions of Tornado or if the Cygwin* Unix-emulator for Windows* platforms is installed.

The instructions for building the IXP400 software include editing and running a script file located in /ixp400_xscale_sw/buildutils. This includes a PATH statement that will manually add the appropriate development tools directories to the systems path environment. However, it may sometimes be desirable to modify these settings according to your particular host platform configuration.

Workaround: If the make command fails, the following steps can be used to troubleshoot the path on the host system:

1. Observe the output from the `make` command. Look for the line at which failures start to appear.

   Typically you will be able to determine which command (for example, `sed` or `make`) is being called from the makefile.

2. In the same command window in which you ran the `make` command, determine which version of the command that caused the make failure is being used. For example: `make --version`.

3. Change directories to the `%WIND_BASE%\host\%WIND_HOST_TYPE%\bin` directory and check the version of the command again.

   If the command is not found in the directory above, you may need to search the system to locate all instances of the command that may be found in the path.

If the versions in Step 2 and 3 are different, the PATH environment on the host platform may need to be modified to resolve the command version conflict.

If the development host system has duplicate copies of certain tools in the host system's path, an incorrect tool version could be called by the makefile during the build process. This is more likely to occur on host systems with multiple versions of Tornado or if the Cygwin* Unix-emulator for Windows* platforms is installed.

2918 RedBoot* May Hang on MontaVista* Linux 3.0 and Earlier, After a Large Transfer on the Flash File System

Issue: The MTD driver sets the Intel StrataFlash in command mode after a write/erase/lock or unlock command is issued. In order to read from the Flash, the IXP4XX has to issue a command for switching the Flash to normal mode. When IXP4XX is reset by the watchdog timer, the Intel StrataFlash® is in command mode, which means IXP4XX is unable to fetch instructions from the flash. This will cause the system will hang.

Workaround: This issue is resolved in MontaVista Linux 3.1.

3239 Incorrect Use of LINUX_VERSION_CODE in 'ixp400_xscale_sw/src/linux/vx.h' Affecting Non-MontaVista* Linux Kernel Compilation

Issue: The build fails when compiling the Intel® IXP400 Software v.1.4 release v1.4 with a non-MVL kernel. This is due to the incorrect use of the LINUX_VERSION_CODE comparison to 2.4.20. Any open source use of the Intel® IXP400 software release v1.4 library will fail to build at the following:

```
~ line 78 of 'ixp400_xscale_sw/src/linux/vx.h':
typedef int clockid_t;
```

This type is for the High Resolution Timer support and is not in 2.4.20.

Workaround: To compile the Intel® IXP400 v1.4 software release with a non-MVL3.1 kernel, there are two options:

- Obtain the High Resolution Timers patch and apply it to the kernel.

  The patch is available from sourceforge at:

  http://sourceforge.net/projects/high-res-timers.

- A temporary workaround — which can be used instead — is to apply the following patch to 'ixp400_xscale_sw/src/linux/vx.h':

```
--- IXP400lib.orig/ixp400_xscale_sw/src/linux/vx.h Mon Dec 15 14:49:24 2003
+++ IXP400lib/ixp400_xscale_sw/src/linux/vx.h Mon Dec 15 21:44:49 2003
@@ -70,13 +70,15 @@

#define CLOCK_REALTIME 0

-#if (LINUX_VERSION_CODE < KERNEL_VERSION(2,4,20))
-/* The Monta Vista 3.1 (2.4.20) kernel headers define a clockid_t
- * type. The following line is for backward compatibility with the 3.0
- * (2.4.18) kernel.
+/* This is added to fixup the dependency on MVL 3.1 having
+ * the high-res-time patch applied to their shipping kernel.
+ * Since a plain-vanilla kernel does not have this, this causes
+ * a build error on any non-mvl3.1 kernel.
+ * See http://high-res-timers.sourceforge.net/
+ *
*/
typedef int clockid_t;
-#endif
+

int clock_gettime(clockid_t clock_id, struct timespec *tp);
```

3251 IXP425_PERIPHERAL_BUS_CLOCK is Incorrectly Defined in Intel® IXP400 Software and VxWorks* BSP as 66 MHz

Issue: The IXP425 Peripheral Bus Clock is incorrectly defined in the VxWorks BSP (ixp425.h) and IXP400 software (`ixOsServices.c`) as `#define IXP425_PERIPHERAL_BUS_CLOCK 66`. The Intel® IXDP425 / IXCDP1100 Development Platform uses a 33.33333-MHz clock. This causes a 1% error to any timing measurements using ticks and timer. Accuracy = 990 µs instead of 1 ms.

This affects all versions of IXP400 software and all versions of the VxWorks IXDP425 / IXCDP1100 platform BSP.

Workaround: Change the definition of IXP425_PERIPHERAL_BUS_CLOCK to:
`#define IXP425_PERIPHERAL_BUS_CLOCK 66.666666`