This document describes the installation and usage of Dolphin PCI Express software stack (eXpressWare) version 5.4.x in combination with Curtiss-Wright PCIe enabled SBCs and backplanes running VxWorks 6.9 or 7.0. Please select the Linux guide for information on installing eXpressWare for Linux.

Publication date February 28th, 2017
Copyright © 2017 Dolphin Interconnect Solutions
Published under Gnu Public License v2
# Table of Contents

Abstract .................................................................................................................. vii
1. Introduction & Overview .................................................................................. 1
   1. Who needs a PCI Express Network? .............................................................. 1
   2. PCI Express networking basics .................................................................. 1
   3. Contents of this Document ......................................................................... 1
   4. Terminology ................................................................................................. 2
   5. Support ......................................................................................................... 3
2. Quick Installation Guide .................................................................................. 4
3. Requirements and Planning ............................................................................ 5
   1. Supported Platforms .................................................................................. 5
      1.1. Application recommendations .............................................................. 5
      1.2. Supported Platforms ......................................................................... 5
      1.3. eXpressWare adapter names ............................................................... 5
      1.4. Adapter serial numbers ...................................................................... 6
      1.5. Recommended Cluster Node Hardware .............................................. 6
   2. Software Support .......................................................................................... 6
      2.1. Linux ..................................................................................................... 6
      2.2. VxWorks .............................................................................................. 6
   3. Interconnect Planning ................................................................................. 7
      3.1. Nodes to Equip with PCI Express Interconnect .................................... 7
      3.2. PCI Express Interconnect Topologies .................................................. 7
4. Initial Installation ............................................................................................. 8
   1. Installation Overview .................................................................................. 8
   2. Software Download .................................................................................... 8
   3. Installing eXpressWare on the development system ................................... 8
   4. Creating an eXpressWare enabled VxWorks kernel ..................................... 11
      4.1. Edit kernel configurations .................................................................. 11
      4.2. Adding core eXpressWare components ............................................. 12
      4.3. Automatic NodeId configuration ........................................................ 12
      4.4. Adding eXpressWare demo Applications .......................................... 13
      4.5. Building the new Kernel Images ......................................................... 14
      4.6. Booting the new kernel ...................................................................... 15
   5. Configuring the Cluster Nodes ................................................................... 15
   6. Verifying the installation ............................................................................ 15
5. Interconnect Maintenance ............................................................................. 17
   1. Verifying Functionality and Performance .................................................. 17
      1.1. Availability of Drivers and Services .................................................. 17
      1.2. Static PCIe Interconnect Test - dis_diag ........................................... 17
      1.3. Interconnect Performance Test ............................................................ 19
   2. Replacing a Node ......................................................................................... 20
   3. Adding Nodes ............................................................................................. 21
6. SISCI API ....................................................................................................... 22
   1. SISCI Documentation and resources ......................................................... 22
   2. Enable applications to use the SISCI API ................................................. 22
   3. How to compile your own SISCI application ............................................ 22
   4. SISCI API Demo and Example programs ................................................... 22
      4.1. SISCI API Example programs ............................................................ 22
         4.1.1. shmem ....................................................................................... 22
         4.1.2. memcpy ....................................................................................... 22
         4.1.3. interrupt .................................................................................... 22
         4.1.4. data_interrupt ........................................................………………….. 23
         4.1.5. intcb ......................................................................................... 23
         4.1.6. lsegcb ....................................................................................... 23
         4.1.7. rsegb ......................................................................................... 23
         4.1.8. dma ......................................................................................... 23
List of Figures

4.1. VxWorks Workbench: Edit Kernel Configurations ................................................................. 12
4.2. VxWorks Workbench: Include DIS eXpressWare Drivers .................................................. 12
4.3. VxWorks Workbench: Automatic NodeId configuration ...................................................... 13
4.4. VxWorks Workbench: eXpressWare applications ............................................................... 13
4.5. VxWorks Workbench: Select demo applications ............................................................... 14
4.6. VxWorks Workbench: Build Project .................................................................................... 14
4.7. VxWorks Workbench: Build Target .................................................................................... 15
List of Tables

3.1. Curtiss-Wright SBC name to eXpressWare adapter mapping .......................................................... 5
4.1. Default automatic NodeId assignment ............................................................... 15
Abstract

This document describes the installation and management of the Dolphins SISCI PCI Express software stack on single machines or on a cluster of machines.

This software stack is needed to use PCI Express for host to host communication and consists of drivers (kernel modules), user space libraries and applications, documentation and more.

The SISCI API provides an easy to use programming environment with direct access to PCI Express features like PIO, DMA, Interrupts and peer to peer transactions.

This document is for clusters built using Curtiss-Wright PCI Express enabled SBCs.
Chapter 1. Introduction & Overview

1. Who needs a PCI Express Network?

Clustered or networked applications running on multiple machines communicating via an Ethernet-based network (1G, 10G, 40G Ethernet) often suffer from the delays that occur when data needs to be exchanged between processes running on different machines. These delays caused by the communication protocols make processes wait for data when they otherwise could perform useful work. Dolphins PCI Express solution is a combination of a high-performance interconnect hardware to replace the Ethernet network together with a highly optimized software stack.

The Dolphin SISCI Developers kit (Software Infrastructure Shared-Memory Cluster Interconnect) consists of driver and API software, tools, documentation and source needed to develop your own embedded application utilizing the low latency and high performance of a PCI Express Cluster. The development kit provides a C system call interface to ease customer integration to Dolphins cluster interconnect. SISCI enables customer application to easily bypass the limitations of traditional network solutions, avoiding time consuming operating system calls, and network protocol software overhead.

The SISCI Software enables applications to directly benefit from remote access time down to 0.54 microseconds and close to wire speed throughput. Direct mapped NTB PIO access gives extremely low transaction overhead and jitter for real-time systems. DMA capabilities provide very high throughput without using CPU for data transfers. Customers can combine PIO and DMA operations for maximum application benefits and performance. The SISCI API also support access to remote IO resources.

2. PCI Express networking basics

The PCI Express network enables a process on one machine to safely write data directly into the address space of a process running on a remote machine. This can be done using either direct store operations of the CPU (for lowest latency), using the DMA engine of the PCI Express chipset or using a system DMA resource (for lowest CPU utilization).

The PCI Express network also enables standard PCI Express devices to be accessed from remote system. It is also possible to enable PCI Express devices to stream data to remote devices or memories.

PCI Express networking is made possible and very flexible by standard Non-Transparent PCI Express bridge functionality that exists in many standard PCI Express chipsets. It enables advanced device drivers to set up direct access to designated communication areas in remote system memory and PCI Express devices.

These hardware features are safely used by Dolphins software to implement several optimized software communication protocols:

The SISCI API implements an easy to use, but yet very powerful API for accessing PCI Express networking transport and functionality directly from applications. It provides a well defined C system call interface to ease customer integration to the PCI Express Network. SISCI enables customer application to easily bypass the limitations of traditional network solutions, avoiding time consuming operating system calls, and network protocol software overhead.

3. Contents of this Document

This document is the installation and reference guide for Dolphins software for PCI Express networking. It is structured as follows:

- Chapter 1, Introduction & Overview is what you are currently reading.

- For those with some experience in cluster management, the abbreviated installation instructions in Chapter 2, Quick Installation Guide might be sufficient. Otherwise, the following chapters cover all details.
Introduction & Overview

- If you are installing the Dolphin PCI Express software stack for the first time, you should refer to Chapter 4, *Initial Installation* for a thorough guide. It explains the cluster installation using the VxWorks Windows Installer.

- Once the cluster is installed, you will find procedures for typical maintenance tasks and for trouble shooting in Chapter 5, *Interconnect Maintenance*.

- Frequently asked questions are answered in Chapter 7, *FAQ*.

- The appendixes provide the configuration files of the Dolphin PCI Express software stack, and a listing of known issues with few platforms and the default software limitations.

### 4. Terminology

We define some terms that will be used throughout this document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adapter</strong></td>
<td>A supported PCI Express cable adapter. This is the PCI Express hardware installed in the Cluster Nodes. Some Cluster Nodes may have an integrated PCIe Express chipset that connects the system to a PCIe enabled backplane. For the sake of this guide, we still in some cases refer to this as an adapter.</td>
</tr>
<tr>
<td><strong>Adapter number</strong></td>
<td>Each PCIe network connections in the SBC is identified by an adapter number. Most SBCs only support one connection</td>
</tr>
<tr>
<td><strong>Cluster Node</strong></td>
<td>A computer which is part of the PCI Express interconnect, which means it has a PCIe Express network connection to other nodes. All Cluster Nodes together constitute the <em>cluster</em>.</td>
</tr>
<tr>
<td><strong>SBC</strong></td>
<td>Single Board Computer (SBC).</td>
</tr>
<tr>
<td><strong>CPU architecture</strong></td>
<td>The CPU architecture relevant in this guide is characterized by the addressing width of the CPU (32 or 64 bit) and the instruction set (x86, PowerPC, Sparc, ARM etc.). If these two characteristics are identical, the CPU architecture is identical for the scope of this guide.</td>
</tr>
<tr>
<td><strong>Fabric</strong></td>
<td>A fabric is an independent, closed communication network that connects a number of machines (here: all nodes in your cluster). Thus, with one adapter in each Cluster Node and all PCIe connections set up, the cluster is using a single fabric.</td>
</tr>
<tr>
<td><strong>Link</strong></td>
<td>The PCIe link between two SBCs or between a SBC and a PCIe switch card.</td>
</tr>
<tr>
<td><strong>Cluster Management Node (frontend)</strong></td>
<td>The single computer that is running software that monitors and configures the Cluster Nodes. The light weight cluster management service communicates with the Cluster Nodes out-of-band, which means via Ethernet. The Cluster Management functionality is not supported or required for systems running VxWorks.</td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
<td>All Cluster Nodes constitute the <em>cluster</em>.</td>
</tr>
<tr>
<td><strong>Network Manager</strong></td>
<td>The Dolphin Network Manager is a daemon process named dis_networkmgr running on the Cluster Management Node. It is part of the Dolphin software stack and manages and controls the <em>cluster</em> using the Node Manager running on all Cluster Nodes. The Network Manager knows the interconnect status of all Cluster Nodes. The service name of the Network Manager is dis_networkmgr.</td>
</tr>
<tr>
<td><strong>Node Manager</strong></td>
<td>The Node Manager is a daemon process that is running on each Cluster Node and provides remote access to the interconnect driver and other Cluster Node...</td>
</tr>
</tbody>
</table>
status to the Network Manager. It reports status and performs actions like configuring the installed adapter.

The Node Manager and Network Manager is not required for VxWorks systems.

VxWorks Windows Installer (MSI)  The VxWorks Windows Installer is an engine for the installation, maintenance, and removal of the VxWorks software on modern Microsoft Windows systems. The installation information, and often the files themselves, are packaged in installation packages, loosely relational databases structured as OLE Structured Storage Files and commonly known as "MSI files", from their default file extension.

IX  The IX keyword is used for specifying a driver that supports PCI Express Gen2 chips from IDT.

PX  The PX keyword is used for specifying a driver that supports PCI Express Gen2 and Gen3 chips from Broadcom/Avago/PLX.

SuperSockets  SuperSockets is a Berkeley sockets compliant socket API provided by Dolphin. SuperSockets is currently supported on systems using Linux and Windows.

SISCI  SISCI (Software Infrastructure for Shared-Memory Cluster Interconnect) is the user-level API to create applications that make direct use of the low level PCI Express interconnect shared memory capabilities.

To run SISCI applications, a service named dis_sisci has to be running; it loads the required kernel module and sets up the SISCI devices.

NodeId  Each Cluster Node in a fabric is identified by an assigned NodeId

x1, x2, x4, x8, x16  PCI express combine multiple lanes (serial high-speed communication channels using few electrical connections) into communication paths with a higher bandwidth. With PCI Express Gen. 1, each lane carries 2.5Gbit/s of traffic, with PCI Express Gen 2, each lane carries 5.0 Gbit/s and with PCI Express Gen3, each lane carries 8.0 Gbit/s. Combining 8 lanes into a single communication path is called x8 and thus delivers 40Gbit/s Bandwidth for Gen 2 or 64Gbit/s Bandwidth for Gen 3, while x16 doubles this bandwidth using 16 lanes and delivers 128Gbit/s for Gen3 in each direction.

5. Support

If you have any problems with the procedures described in this document, please contact Curtiss-Wright support https://www.curtisswrightds.com/support.html.

Please contact Dolphin via <sisci-support@dolphinics.com> if you need assistance on developing your own SISCI application. Dolphin currently offers several levels of SISCI Developer support programs.
Chapter 2. Quick Installation Guide

This chapter gives an overview of the steps necessary for a default installation of Dolphins eXpressWare for VxWorks using the VxWorks Windows Installer (MSI).

Please consult the detailed installation guide for more details, Chapter 4, Initial Installation.

1. **Installation requirements:** The eXpressWare installation requires a 32 or 64 bit Windows system with the required VxWorks target BSPs pre-installed. The installation requires around 5 Megabytes of free storage space.

   For details, see Chapter 3, Requirements and Planning for general platform requirements of the Dolphin PCI Express software stack and Chapter 4, Initial Installation, Section 1, “Installation Overview” for specific requirements for this installation procedure.

2. **Software download:** The Dolphin eXpressWare software for Curtiss-Wright systems is provided by Curtiss-Wright on a CD. Please copy the VxWorks Windows installer to a directory on the machine that will serve as the development system. Make sure root has write access to this directory.

   Please contact Curiss-Wright support if you need help to identify the required software: https://www.curtisswrightds.com/support.html

3. **Start software installation:** Launch the MSI package.

   **Note**

   Please carefully read the licensing terms and install the software only if you accept. Navigate to the pre-installed BSP and install all Dolphin eXpressWare inside the BSP directory.

   The Dolphin eXpressWare usage license requires the user to purchase a run time license for each system (PC, SBC etc) running the software. (e.g. To run a 4 node system, you need to purchase 4 licenses.)

   Please verify you have purchased the required number of licenses or contact Curtiss-Wright sales to acquire the required number of licenses.

4. **Building the target kernel:** Select the appropriate Dolphin Express components (CDF) and build your new target kernel using WindRiver Workbench. Install the new eXpressWare enabled kernels on all target systems.

5. **Interconnect configuration:** After the new kernels has been installed, you need to configure each Cluster Node. Each system must be assigned an unique NodeId. This is done in the script loading the kernel modules.

6. **Verify the installation:** After booting the eXpressWare enabled VxWorks kernel on all systems, please run dis_diag on all nodes to verify the installation and connectivity

   **Warning**

   Please carefully perform this test. If you enable your application to use PCI Express without completing this step, you may be required to stop or restart your application to overcome possible installation issues at a later stage. All tests should pass before using the PCI Express cluster for production purposes. At this point, you can safely execute your tests without affecting the system.

7. **Reboot the system:** It is recommended to reboot the system to verify the automatic restart of the drivers and to ensure optimal memory allocation for the PCI Express drivers.

8. **Run you application:** At this point you can start your SISCI application and it will be able to use PCI Express communication.

   The Dolphin software installation comes with several example, demo and benchmark programs that can be executed to determine the actual PCI Express PIO, DMA and Interrupt performance.
Chapter 3. Requirements and Planning

Before you deploy a PCI Express cluster solution by either adding it to an existing system, or by planning it into a new system, some considerations on selection of products and the physical setup are necessary.

1. Supported Platforms

The Dolphin PCI Express software stack used with Curtiss-Wright systems is available for VxWorks 6.9 / 7 and Linux.

1.1. Application recommendations

PCI Express should normally be used between all systems that requires low latency or high throughput.

Most systems will provide low latency for small amount of data. Applications that needs high throughput will normally benefit from selecting a platform that provides DMA capabilities. PCI Express Gen3 based platforms are recommended for highest throughput.

1.2. Supported Platforms

The Dolphin PCI Express software is supported on a wide range of Curtiss-Wright Single Board Computers (SBCs) and Digital Signal Processor (DSP) modules, and supports any PCIe lane width and bus speed interconnect. For modules that support on-board DMA, the libraries have been optimized to make use of DMA to maximize data throughput and reduce CPU overhead. Supported Curtiss-Wright modules include:

- VPX3-1258: Intel 4th Gen Core i7 “Haswell” SBC
- VPX3-1259: Intel 5th Gen Core i7 “Broadwell” SBC
- VPX3-482 (Champ XD1): Intel Xeon D DSP Engine
- VPX3-131: NXP P4080 Power Architecture SBC
- VPX3-133: NXP T2080 Power Architecture SBC

Some combinations of CPU and chipset implementations offer sub-optimal performance which should be considered when planning a new system.

If you have questions about your specific hardware platform, please compare with the known issues listed in Appendix B, Platform Issues and Software Limitations or contact support.

1.3. eXpressWare adapter names

The table below gives the mapping between Curtiss-Wright SBC part numbers and the adapter names used by the eXpressWare. Adapter names are used by the eXpressWare drivers to identify and manage the underlying PCI Express chipset.

<table>
<thead>
<tr>
<th>Curtiss-Wright SBC name</th>
<th>eXpressWare Adapter name</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPX3-1258</td>
<td>PCH258</td>
</tr>
<tr>
<td>VPX3-1259</td>
<td>PCH259</td>
</tr>
<tr>
<td>VPX3-482</td>
<td>PCH482</td>
</tr>
<tr>
<td>VPX3-133</td>
<td>PCH133</td>
</tr>
</tbody>
</table>
### 1.4. Adapter serial numbers

The eXpressWare software will automatically assign each SBC a serial number that can be reached over PCIe to determine the remote partner.

The PCIe serial number for an SBC is calculated from the MAC address. If the MAC address is `aa:bb:cc:dd:ee:ff` the serial number is calculated as using the following formula:

\[
\text{serial_no} = \text{ddeeфф(hex)} \mod 1000000(\text{dec})
\]

For example:

MAC 00:1b:ac:00:90:6a  
serial_no = 0x00906a modulo 1000000 = 36970

### 1.5. Recommended Cluster Node Hardware

The hardware platform for the Cluster Nodes should be chosen from the Supported Platforms as described above. Next to the PCI Express specific requirements, you need to consult your application vendor expert / consultant on the recommended configuration for your application.

The PCI Express interconnect is fully inter-operable between all supported single board computers except when using the VPX3-131. (Please consult the release note for details.) As with all applications that communicate over a network, care must be taken by the applications if data with different endianess is communicated.

### 2. Software Support

The Dolphin PCI Express eXpressWare software stack running on Curtiss-Wright systems is currently available for Linux and VxWorks.

#### 2.1. Linux

The complete Dolphin PCI Express software stack can be compiled for all popular Linux distributions (check the release note for details). A few extra packages (like the kernel include files and configuration) need to be installed for the compilation. Dolphin distributes an installer package that contains both binaries and source that needs to be compiled on the target system. Software stacks operating on different kernel versions are of course fully inter-operable for inter-node communication.

Dolphin PCI Express software fully supports native 32-bit and 64-bit platforms. On 64-bit platforms offering both 32-bit and 64-bit runtime environments, SuperSockets will support 32-bit applications if the compilation environment for 32-bit is also installed. Otherwise, only the native 64-bit runtime environment is supported.

For more information refer to Chapter 7, FAQ.

Please refer to the release notes of the software stack version you are about to install for the current list of tested Linux distributions and kernel versions. Installation and operation on Linux distributions and kernel versions that are not in this list will usually work as well, but especially the most recent versions may cause problems if it has not yet been qualified by Dolphin. Please notify Dolphin support at `<pci-support@dolphinics.com>` if you have a problem.

#### 2.2. VxWorks

The Dolphin PCI Express SISCI software stack is available for VxWorks 6.9 and VxWorks 7. Please consult the software release note for details.
3. Interconnect Planning

This section discusses the decisions that are necessary when planning to install a PCI Express cluster.

3.1. Nodes to Equip with PCI Express Interconnect

Depending on the application that will run on the cluster, the choice of PCI Express Interconnect equipped machines differs.

3.2. PCI Express Interconnect Topologies

Two Cluster Nodes can be connected directly through a backplane.

Three Cluster Nodes can be connected directly through the backplane using the multi port topology.

Four and five Cluster Nodes can be connected directly through the backplane using the multi port topology, but only neighbour nodes will be able to communicate.

Up to 8 Cluster Nodes can be connected to one central PCIe switch.
Chapter 4. Initial Installation

This chapter guides you through the initial hardware and software installation of eXpressWare for VxWorks. This means that no Dolphin software is installed on the Cluster Nodes or the Cluster Management Node prior to these instructions.

The recommended installation procedure, which is described in this chapter, uses the VxWorks Windows installer image *.msi distribution of the Dolphin PCI Express software stack.

1. Installation Overview

*Installation requirements:* The installation requires a Windows system with the required VxWorks target BSPs pre-installed.

2. Software Download

The Dolphin eXpressWare software for Curtiss-Wright systems is provided by Curtiss-Wright on a CD. Please download/copy the VxWorks Windows installer to a directory on the machine that will serve as the development system. Make sure root has write access to this directory.

3. Installing eXpressWare on the development system

Installing the eXpressWare on the Windows development system is easy. Please follow the steps below:

1. Launch the MSI package by double clicking on the installer icon. Please make sure you have the appropriate installer for your BSP/SBC.

   ![eXpressWare for VxWorks Setup](image)

   New users of eXpressWare are encouraged to read the installation and usage documentation before proceeding. Please click on the "View Documentation" button to visit the online documentation.

   The documentation is also available in pdf format.

2. Please accept to install the driver, utilities, benchmarks, sample programs, header files and documentation for Dolphin eXpressWare. The default selection is recommended except for the installation folder. By default the
default folder is set to C:\WindRiver\ (which is the normal default Wind River installation base on Windows development station).

Use the "Browse" button to navigate to the pre-installed BSP and install all Dolphin eXpressWare inside the BSP directory. For example, if you are installing the SW for the VPX3-131 VxWorks 7 BSP, browse the BSP folder "C:\WindRiver\vxworks-7\pkgs\os\board\ppc\cw_131-1.0.0.1".

The installer will install the binary eXpressWare archive directly in the selected BSP directory.

The installer will also install a directory that contains SISCI header files needed to compile your own application using the SISCI API and several SISCI demo and example programs.

3. Please carefully read the licensing terms and install the software only if you accept.

Note

Please note that you are required to purchase a runtime license for each systems running the xPressWare software stack unless you also have a Dolphin adapter card installed.

Please verify you have purchased the required number of licenses or contact Curtiss-Wright sales to acquire the required number of licenses.
4. Click on "Install" to install the software.

5. The User Account Control window will pop up. Please click "Yes" to allow this app to install software on your PC".

6. After a successful installation, please click "Finish" to exit the installer.
7. After the installation completes, you will find the installed files in two directories. If you installed VxWorks 7 software for the VPX3-131 BSP, the directories are:

C:\WindRiver\vxworks-7\pkgs\os\board\ppc\cw_131-1.0.0.1
C:\WindRiver\vxworks-7\pkgs\os\board\ppc\cw_131-1.0.0.1_demo

Similar, if you installed VxWorks 6.9 software for the VPX3-131 BSP, the directories are:

C:\WindRiver\vxworks-6.9\target\config\cwv131r120
C:\WindRiver\vxworks-6.9\target\config\cwv131r120\src\demo

4. Creating an eXpressWare enabled VxWorks kernel

4.1. Edit kernel configurations

To be able to create a new VxWorks kernel image from a Curtiss-Wright BSP you need an updated VxWorks Source Build (VSB) project, i.e., the project must be based on a BSP that includes Dolphin eXpressWare drivers. Please refer to the WindRiver documentation for instructions on how to create a VSB project based on a BSP.
Initial Installation

Figure 4.1. VxWorks Workbench: Edit Kernel Configurations

Based on this updated VSB project, a VxWorks Image Project (VIP) must be created. As seen in Figure 4.1, "VxWorks Workbench: Edit Kernel Configurations", the first step to adding eXpressWare PCIe drivers is to edit the "Kernel Configuration". This is achieved in Workbench by right-clicking on your VIP project (in Project Explorer) and then choosing "Edit Kernel Configuration".

4.2. Adding core eXpressWare components

Figure 4.2. VxWorks Workbench: Include DIS eXpressWare Drivers

By choosing "Edit Kernel Configuration", a new editor tab is created, see Figure 4.2, “VxWorks Workbench: Include DIS eXpressWare Drivers”. In this tab, locate the folder "DIS eXpressWare Software Components", found in the "BSP Device Drivers" folder.

By right-clicking on the folder and choosing "Include (Quick)", the default component ("DIS eXpressWare Drivers") is included. Note that including a component should automatically highlight its name in bold, see Figure 4.3, “VxWorks Workbench: Automatic NodeId configuration”.

4.3. Automatic NodeId configuration

The recommended way to configure the PCI Express cluster is by using the automatic NodeId assignment.
The eXpressWare driver and its user applications require an unique NodeId to be assigned to every compute node in a cluster. This NodeId can be automatically configured based on which chassis slot the Curtiss-Wright board is mounted into.

To enable this auto-configuration, include the "DIS eXpressWare Node Auto ID" component as shown in Figure 4.3, “VxWorks Workbench: Automatic NodeId configuration”. More information on Cluster configuration can be found in Section 5, “Configuring the Cluster Nodes”

4.4. Adding eXpressWare demo Applications

Select the appropriate SISCI demo and benchmark applications to be included.

To add examples and benchmarking applications to the kernel image, right click on the "DIS eXpressWare Applications" folder (Figure 4.4, “VxWorks Workbench: eXpressWare applications”). This brings up a pop-up window, see Figure 4.5, “VxWorks Workbench: Select demo applications”, from which you can choose the applications to include. When you happy with your selection, click "Finish".
4.5. Building the new Kernel Images

After adding eXpressWare drivers and applications, you need to build the new kernel image. As shown in Figure 4.6, “VxWorks Workbench: Build Project” below, this is done by right-clicking the VIP project and choosing "Build Project” and then building the preferred kernel target. E.g. Figure 4.7, “VxWorks Workbench: Build Target” below shows how to build a U-Boot image. Note that the new kernel image must then be made available for the target during boot.

Figure 4.6. VxWorks Workbench: Build Project
4.6. Booting the new kernel

Boot all target systems using the new VxWorks kernel

5. Configuring the Cluster Nodes

Each node in a PCI Express cluster is identified by a system NodeId. The system NodeId must be unique and a multiple of 4 (e.g. 4, 8, 12).

The Cluster Nodes can be configured automatically by defining INCLUDE_DIS_AUTO_CONFIG when building the kernel. Auto configuration is based on the slot number, the NodeId will be assigned based on the slot position:

Table 4.1. Default automatic NodeId assignment

<table>
<thead>
<tr>
<th>Slot</th>
<th>NodeId</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>x</td>
<td>(x+1)*4</td>
</tr>
</tbody>
</table>

The NodeId assignments can be changed for a slot by editing the dis_init.c file. The NodeId must be unique in the cluster, and always a multiple of 4. The maximum NodeId is currently 32.

If the INCLUDE_DIS_AUTO_CONFIG is not defined, each adapter must be configured manually. E.g.:

```
sp(dis_config, "-n 4")
```

where the argument after '-n' is the NodeId.

6. Verifying the installation

It is recommended to reboot all Cluster Nodes to verify the automatic restart of the drivers and to ensure optimal memory allocation for the PCI Express drivers.
After the reboot, please run dis_diag on all Cluster Nodes to verify the installation and connectivity.
Chapter 5. Interconnect Maintenance

This chapter describes how to perform a number of typical tasks related to the PCI Express interconnect.

1. Verifying Functionality and Performance

The following sections describe how to verify that the interconnect is setup correctly, which means that all Cluster Nodes can communicate with all other Cluster Nodes via the PCI Express interconnect by sending low-level control packets and performing remote memory access.

1.1. Availability of Drivers and Services

Without the required drivers and services running on a Cluster Node, the node will fail to communicate with other nodes. On the Cluster Nodes, the kernel services (interconnect resources driver) and (upper level hardware services) need to be running.

Because the drivers do also appear as services, you can query their status with the usual tools of the installed operating system distribution.

If any of the required services is not running, you will find more information on the problem that may have occurred in the system log facilities. Call to inspect the kernel messages, and check for related messages.

1.2. Static PCIe Interconnect Test - dis_diag

The static interconnect test makes sure that all PCIe communication hardware are working correctly by performing a self-test, and determines if the setup and the PCIe routing is correct (matches the actual hardware topology). It will also check all PCIe connections, but this has already been done in the PCIe Connection Test. The tool to perform this test is dis_diag (default location).

Running dis_diag on a Cluster Node will perform a self test on the local adapter(s) and list all remote adapters that this adapter can see via the PCI Express interconnect. This means, to perform the static interconnect test on a full cluster, you will basically need to run dis_diag on each Cluster Node and see if any problems with the adapter are reported, and if the adapters in each Cluster Node can see all remote adapters installed in the other Cluster Nodes.

Normally you should invoke dis_diag with no arguments, and it will do a general test and only show the most interesting information. Advanced users may want to enable the full verbose mode by using the -V 9 command line option:

```
$ sp(dis_diag "-V 9")
```

The -V 9 option will generate a lot of information, some parts of the information requires knowledge about the PCIe chipset and the PCIe specification in general. The diagnostic module will collect various usage and error information over time. This can be cleared by using the -clear command line option:

```
$ sp(dis_diag "-clear")
```

An example output of dis_diag for a Cluster Node which is part of a 3 node Asymmetric cluster topology looks like this:

```
Dolphin diagnostic tool --  dis_diag version 5.4.2 ( Mon Jan 23 18:04:18 CET 2017 )

Driver : Dolphin IRM (GX) 5.4.2.0 Jan 20th 2017 (rev 828884a)
Date    : Not available
System  : Not available
```
Number of configured local adapters found: 1

Adapter 0 > Type                       : PCH133
    NodeId                     : 8
    Serial number              : PCH133-AA-039113
    PXH chip family            : PLX_SIRIUS
    PXH chip vendorId          : 0x10b5
    PXH chip device            : 0x8619
    PXH chip revision          : 0xBA
    EEPROM version NTB mode    : 02
    EEPROM vendor info         : 0x0000
    Card revision              : AA
    Topology type              : Asymmetric
    Topology Autodetect        : No
    Number of enabled links    : 2
    Max payload size (MPS)     : 128
    Multicast group size       : 2 MB
    Non-prefetchable memory size : 256 MB (BAR2)
    Clock mode upstream        : Port
    Clock mode link            : Global
    PCIe upstream state        : x8, Gen2 (5 GT/s)
    PCIe upstream capabilities : x8, Gen2 (5 GT/s)

*************************  PXH ADAPTER 0 PCIe0 STATE *************************
    PCIe0 uptime               : 98 seconds
    PCIe0 state                : ENABLED
    PCIe0 state                : x4, Gen2 (5 GT/s)
    PCIe0 configured           : x4, Gen2 (5 GT/s)
    PCIe0 capability           : NA
    PCIe0 active               : 1
    PCIe0 configuration        : TRANSPARENT

*************************  PXH ADAPTER 0 PCIe1 STATE *************************
    PCIe1 uptime               : 98 seconds
    PCIe1 state                : ENABLED
    PCIe1 state                : x4, Gen2 (5 GT/s)
    PCIe1 configured           : x4, Gen2 (5 GT/s)
    PCIe1 capability           : x4, Gen2 (5 GT/s)
    PCIe1 active               : 1
    PCIe1 configuration        : NTB

*****************************  TEST OF ADAPTER 0 ****************************
    OK: PXH chip alive in adapter 0.
    OK: Link alive in adapter 0.
    ==> Local adapter 0 ok.

****************  PXH ADAPTER 0, PARTNER INFORMATION FOR PCIe0 ****************
    Partner information not available

****************  PXH ADAPTER 0, PARTNER INFORMATION FOR PCIe1 ****************
    Partner information not available

***********************  SESSION STATUS FROM ADAPTER 0 ***********************
    Node 8: Session valid
    Node 12: Session valid

Adapters found: 3
----- List of all nodes found:
Nodes detected: 0008 0012 0016
The static interconnect test passes if dis_diag delivers TEST RESULT: *PASSED* and reports the same topology (remote adapters) on all Cluster Nodes.

### 1.3. Interconnect Performance Test

Once the correct installation and setup and the basic functionality of the interconnect have been verified, it is possible to perform a set of low-level benchmarks to determine the base-line performance of the interconnect without any additional software layers. The tests that are relevant for this are scibench2 (streaming remote memory PIO access performance), scipp (request-response remote memory PIO write performance), dma_bench (streaming remote memory DMA access performance) and intr_bench (remote interrupt performance).

All these tests need to run on two Cluster Nodes (A and B) and are started in the same manner:

1. Determine the NodeId of both Cluster Nodes using the `query` command (default path). The NodeId is reported as "Local node-id".

2. On node A, start the server-side benchmark with the options `-server` and `-rn <NodeId of B>`, like:
   
   ```sh
   $ sp(scibench2 "-server -rn 8")
   ```

3. On Cluster Node B, start the client-side benchmark with the options `-client` and `-rn <NodeId of A>`, like:
   
   ```sh
   $ sp(scibench2 "-client -rn 4")
   ```

4. The test results are reported by the client.

scibench2

Scibench2 measures the streaming bandwidth using CPU based PIO transfers (memcpy)

The following results are measured using a PCI Express Gen3 x8 link

<table>
<thead>
<tr>
<th>Segment Size</th>
<th>Average Send Latency</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.07 us</td>
<td>58.31 MBytes/s</td>
</tr>
<tr>
<td>8</td>
<td>0.07 us</td>
<td>117.14 MBytes/s</td>
</tr>
<tr>
<td>16</td>
<td>0.07 us</td>
<td>231.06 MBytes/s</td>
</tr>
<tr>
<td>32</td>
<td>0.08 us</td>
<td>445.08 MBytes/s</td>
</tr>
<tr>
<td>64</td>
<td>0.08 us</td>
<td>838.84 MBytes/s</td>
</tr>
<tr>
<td>128</td>
<td>0.09 us</td>
<td>1483.27 MBytes/s</td>
</tr>
<tr>
<td>256</td>
<td>0.11 us</td>
<td>2408.40 MBytes/s</td>
</tr>
<tr>
<td>512</td>
<td>0.15 us</td>
<td>3497.44 MBytes/s</td>
</tr>
<tr>
<td>1024</td>
<td>0.23 us</td>
<td>4530.20 MBytes/s</td>
</tr>
<tr>
<td>2048</td>
<td>0.39 us</td>
<td>5294.99 MBytes/s</td>
</tr>
<tr>
<td>4096</td>
<td>0.77 us</td>
<td>5308.03 MBytes/s</td>
</tr>
<tr>
<td>8192</td>
<td>1.54 us</td>
<td>5306.69 MBytes/s</td>
</tr>
<tr>
<td>16384</td>
<td>3.10 us</td>
<td>5291.49 MBytes/s</td>
</tr>
<tr>
<td>32768</td>
<td>6.19 us</td>
<td>5294.48 MBytes/s</td>
</tr>
<tr>
<td>65536</td>
<td>12.39 us</td>
<td>5289.90 MBytes/s</td>
</tr>
</tbody>
</table>

**Average Send latency** is the wall time to write 4 bytes to remote memory

**Throughput** is the streaming performance using PIO writes to remote memory.

The actual performance will depend on your PCIe link capabilities, x4, x8, Gen1, Gen2, Gen3 etc.
dma_bench

dma_bench measures the streaming DMA bandwidth available through the SISCI API.

The following results are measured using a PCI Express Gen3 x16 link and a PLX chip DMA engine.

<table>
<thead>
<tr>
<th>Message size</th>
<th>Total size</th>
<th>Vector length</th>
<th>Transfer time</th>
<th>Latency per message</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>16384</td>
<td>256</td>
<td>35.76 us</td>
<td>0.14 us</td>
<td>458.18 MBytes/s</td>
</tr>
<tr>
<td>128</td>
<td>32768</td>
<td>256</td>
<td>36.81 us</td>
<td>0.14 us</td>
<td>890.24 MBytes/s</td>
</tr>
<tr>
<td>256</td>
<td>65536</td>
<td>256</td>
<td>37.16 us</td>
<td>0.15 us</td>
<td>1763.43 MBytes/s</td>
</tr>
<tr>
<td>512</td>
<td>131072</td>
<td>256</td>
<td>39.36 us</td>
<td>0.15 us</td>
<td>3329.83 MBytes/s</td>
</tr>
<tr>
<td>1024</td>
<td>262144</td>
<td>256</td>
<td>41.34 us</td>
<td>0.16 us</td>
<td>6340.40 MBytes/s</td>
</tr>
<tr>
<td>2048</td>
<td>524288</td>
<td>256</td>
<td>54.75 us</td>
<td>0.21 us</td>
<td>9576.21 MBytes/s</td>
</tr>
<tr>
<td>4096</td>
<td>524288</td>
<td>128</td>
<td>51.83 us</td>
<td>0.40 us</td>
<td>10116.51 MBytes/s</td>
</tr>
<tr>
<td>8192</td>
<td>524288</td>
<td>64</td>
<td>50.46 us</td>
<td>0.79 us</td>
<td>10390.38 MBytes/s</td>
</tr>
<tr>
<td>16384</td>
<td>524288</td>
<td>32</td>
<td>49.69 us</td>
<td>1.55 us</td>
<td>10551.60 MBytes/s</td>
</tr>
<tr>
<td>32768</td>
<td>524288</td>
<td>16</td>
<td>49.30 us</td>
<td>3.08 us</td>
<td>10537.61 MBytes/s</td>
</tr>
<tr>
<td>65536</td>
<td>524288</td>
<td>8</td>
<td>49.07 us</td>
<td>6.13 us</td>
<td>10684.71 MBytes/s</td>
</tr>
<tr>
<td>131072</td>
<td>524288</td>
<td>4</td>
<td>48.90 us</td>
<td>12.23 us</td>
<td>10721.20 MBytes/s</td>
</tr>
<tr>
<td>262144</td>
<td>524288</td>
<td>2</td>
<td>48.89 us</td>
<td>24.44 us</td>
<td>10724.27 MBytes/s</td>
</tr>
<tr>
<td>524288</td>
<td>524288</td>
<td>1</td>
<td>48.98 us</td>
<td>48.98 us</td>
<td>10704.78 MBytes/s</td>
</tr>
</tbody>
</table>

The actual performance will depend on your PCIe link capabilities, x4, x8, Gen1, Gen2, Gen3 etc, type of DMA (PCIe chip or system DMA).

DMA is not supported by all SBCs. Please consult the SBC documentation for more details.

scipp

The scipp SISCI benchmark sends a message of the specified size to the remote system. The remote system is polling for incoming data and will send a similar message back to the first node.

The minimal round-trip latency for writing to remote memory is extremely low using PCI Express networks.

The following results are typical for a PCI Express Gen3 x8 link

<table>
<thead>
<tr>
<th>Ping Pong data transfer:</th>
<th>retries</th>
<th>latency (usec)</th>
<th>latency/2 (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2486</td>
<td>1.079</td>
<td>0.539</td>
</tr>
<tr>
<td>4</td>
<td>2406</td>
<td>1.078</td>
<td>0.539</td>
</tr>
<tr>
<td>8</td>
<td>2442</td>
<td>1.090</td>
<td>0.545</td>
</tr>
<tr>
<td>16</td>
<td>2454</td>
<td>1.098</td>
<td>0.549</td>
</tr>
<tr>
<td>32</td>
<td>2482</td>
<td>1.117</td>
<td>0.558</td>
</tr>
<tr>
<td>64</td>
<td>2562</td>
<td>1.151</td>
<td>0.575</td>
</tr>
<tr>
<td>128</td>
<td>2608</td>
<td>1.176</td>
<td>0.598</td>
</tr>
<tr>
<td>256</td>
<td>2667</td>
<td>1.247</td>
<td>0.624</td>
</tr>
<tr>
<td>512</td>
<td>2866</td>
<td>1.331</td>
<td>0.666</td>
</tr>
<tr>
<td>1024</td>
<td>3064</td>
<td>1.492</td>
<td>0.746</td>
</tr>
<tr>
<td>2048</td>
<td>3773</td>
<td>1.880</td>
<td>0.940</td>
</tr>
<tr>
<td>4096</td>
<td>4850</td>
<td>2.659</td>
<td>1.330</td>
</tr>
<tr>
<td>8192</td>
<td>7364</td>
<td>4.247</td>
<td>2.123</td>
</tr>
</tbody>
</table>

intr_bench

The interrupt latency is affected by the operating system and can therefore vary.

Average unidirectional interrupt time : 2.515 us.
Average round trip interrupt time : 5.030 us.

2. Replacing a Node

In case a Cluster Node needs to be replaced, proceed as follows concerning the PCIe interconnect:
Interconnect Maintenance

1. Power down the Cluster Node.

2. Replace the selected SBC and make sure the correct firmware is installed.

3. Power up the SBC.

4. Run dis_diag to verify the functionality of the interconnect (see Section 1.2, “Static PCIe Interconnect Test - dis_diag”).

Communication between all other Cluster Nodes will continue uninterrupted during this procedure.

3. Adding Nodes

To add new Cluster Nodes to the cluster, please proceed as follows:

1. Install the new SBC in the backplane and power it up. Ensure the appropriate PCIe switch firmware is installed on the SBC.

2. Install the VxWorks kernel that contains the Dolphin eXpressWare software stack on all Cluster Nodes.

3. Run dis_diag to verify the functionality of the interconnect (see Section 1.2, “Static PCIe Interconnect Test - dis_diag”).

4. Now you can start the applications on the new Cluster Nodes as you did on the old Cluster Nodes, and they will be able to connect to the old Cluster Nodes.
Chapter 6. SISCI API

This chapter explains how to use the SISCI API.

SISCI is a powerful remote/shared memory API that makes it easy to write an application that can make local resources available to remote systems and access remote resources using PCI Express. The SISCI API can be used to implement various types of interprocess communication and message passing.

1. SISCI Documentation and resources

The SISCI API specification and the SISCI Users Guide can be found at http://www.dolphinics.com/products/embedded-sisci-developers-kit.html

The eXpressWare installation for all supported operating systems comes with several SISCI example, demo and benchmark programs. The source code is included.

2. Enable applications to use the SISCI API

Applications needs to be written following the SISCI API specification to be able to use SISCI enabled communication.

SISCI uses the PCI Express interconnect for low-latency, high-bandwidth communication inside the cluster.

To run a SISCI application on a Cluster Node, the SISCI library and the eXpressWare drivers must be compiled into the running kernel and configured properly.

3. How to compile your own SISCI application

To compile and link SISCI application, you need various SISCI header files from the eXpressWare installation.

4. SISCI API Demo and Example programs

The eXpressWare development package comes with several example and benchmark programs. It is recommended to study these before designing your own application.

All SISCI example, demo and benchmark programs supports various command line options, details will be provided during runtime if you start each application with the -help option.

4.1. SISCI API Example programs

The purpose of the example programs is to demonstrate the basic usage of selected SISCI API functionality.

4.1.1. shmem

The shmem program code demonstrates how to create a basic SISCI program and exchange data using PIO. An interrupt is created and signalled when the data exchange is completed.

4.1.2. memcpy

The memcpy program code demonstrates how to create a basic SISCI program and exchange data using PIO. An interrupt is created and signalled when the data exchange is completed.

4.1.3. interrupt

The interrupt program code demonstrates how to trigger an interrupt on a remote system using the SISCI API. The receiver thread is blocking, waiting for the interrupt to arrive.
4.1.4. **data_interrupt**
The data_interrupt program code demonstrates how to trigger an interrupt with data on a remote system.

4.1.5. **intcb**
The intcb program code demonstrates how to trigger an interrupt on a remote system. The receiver thread is notified using an interrupt callback function.

4.1.6. **lsegcb**
The lsegcb program code demonstrates the use of local segment callbacks.

4.1.7. **rsegcb**
The rsegcb program code demonstrates the use of remote segment callbacks.

4.1.8. **dma**
The dma program code demonstrates the basic use of DMA operations to move data between segments.

4.1.9. **dmacb**
The dma program code demonstrates the basic use of DMA operations to move data between segments using the completion callback mechanism.

4.1.10. **dmavec**
The dma program code demonstrates how to set up a vectorized DMA operations.

4.1.11. **rpcia**
The rpcia program code demonstrates how to use the PCIe peer to peer functionality to enable remote systems to access a local PCIe resource / Physical address within the system.

4.1.12. **reflective_memory**
The reflective_memory program code demonstrates how to use PCIe multicast / reflective memory functionality. Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.13. **reflective_dma**
The reflective_dma program code demonstrates how to use multicast / reflective memory with DMA transfers. Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.14. **reflective_device**
The reflective_device program code demonstrates how to use the SISCI API to enable a PCIe device to directly utilize the PCIe multicast / reflective memory functionality. Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.

4.1.15. **reflective_write**
The reflective_write program code demonstrates how to use PCIe multicast / reflective memory functionality. Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.
4.1.16. probe

The probe program code demonstrates how to determine if a remote system is accessible via the PCIe network.

4.1.17. query

The query program code demonstrates how to identify various system properties and status settings.

4.2. SISCI API demo and benchmarks programs

The purpose of the benchmark and demo programs is to demonstrate how to measure the actual communication performance over the PCIe network.

The source for all the benchmark and demo programs will be installed when you install the eXpressWare components.

4.2.1. scibench2

The scibench2 program can be used to determine the actual CPU load/store performance to a remote or local segment.

The program copies data to the remote segment without any synchronization between the client and server side during the benchmark.

The send latency displayed by the application is the wall clock time to send the data once.

4.2.2. scipp

The scipp program can be used to determine the actual CPU store latency to a remote or local segment.

The program will sends data to the remote system. The remote system is polling for new data and will send a similar amount of data back when it detects the incoming message.

4.2.3. dma_bench

The dma_bench program can be used to determine the actual DMA performance to a remote or local segment.

The program connects to a remote segment and executes a series of single sided DMA operations copying data from a local segment to a remote segment. There is no synchronization between the client and server side during the benchmark.

DMA operations is not supported by all systems. The program will fail if the local system does not support PCIe chip or System DMA.

4.2.4. intr_bench

The intr_bench program can be used to determine the actual latency for sending a remote interrupt.

The program implements a interrupt ping - pong benchmark where the client and server sides exchanges interrupts and measures the full round trip latency. The interrupt latency measured by the program will be the average of both systems. The interrupt latency measured is the full application to application latency.

4.2.5. reflective_bench

The reflective_bench program can be used to benchmark the reflective memory / multicast functionality enabled by PCI Express networks.

The program implements a multicast data ping - pong benchmark where the client and server sides exchanges multicast data.

Reflective memory functionality is fully supported in two node configurations and with a central switch.
Please note that the 3, 4, 5 node Asymmetric topology does not support PCIe multicast.
Chapter 7. FAQ

This chapter contains some frequently asked questions and answers.

1. Software

1.1.1. The eXpressWare drivers (IRM etc) fails to load after it has been installed for the first time.

   Please follow the procedure below to determine the cause of the problem.

   1. Verify that the PCIe chip has been recognized by the machine and that the appropriate firmware has been loaded.

   2. Check the kernel message log for relevant messages.

   3. If the driver still fails to load, please contact support and provide the driver's kernel log messages.

1.1.2. Is the Network Manager and Node Manager used on VxWorks?

   No, these utilities are not supported and is not needed to run SISCI applications under VxWorks. The configuration and management of real-time systems are made static and manual to avoid running tasks that may disturb real-time applications.

1.1.3. Is SuperSockets and the TCP/IP driver available on VxWorks?

   No, SuperSockets and the TCP/IP driver is currently only available with Linux and Windows, please contact Dolphin for more information.
Appendix A. Configuration Files

1. Cluster Configuration

eXpressWare for VxWorks is currently not configured using the dishosts.conf configuration file. Special care must be taken to make sure all nodes in a PCI Express network have unique NodeIds. Other nodes running other operating systems, configured via the dishosts.conf file, should not list the VxWorks nodes and NodeIds.
Appendix B. Platform Issues and Software Limitations

This chapter lists known issues of PCI Express with certain hardware platforms and limitations of the software stack. Some of these limitations can be overcome by changing default settings of runtime parameters to match your requirements.

1. Platforms with Known Problems vs Dolphin PCI Express software

   Low PIO performance

   PowerPC systems does not support CPU write combining. The effect of this is that every CPU store will be transmitted to the remote memory as a small PCIe transfer. This will normally have a huge impact on the write performance compared to e.g. an Intel CPU running at the same speed. PowerPC users should use DMA operations to transfer larger amounts of data.

2. IRM

   Resource Limitations

   The IRM (Interconnect Resource Manager) manages the hardware and related software resources of the PCI Express interconnect. Some resources are allocated once when the IRM is loaded. The default setting are sufficient for typical cluster sizes and usage scenarios. Please contact Dolphin support if you hit a resource limitation.

3. SISCI

   Please consult the software release note for details.

   Heterogeneous Cluster Operation (Endianess)

   SISCI supports clusters where Cluster Nodes use different endian representation (mixing little endian or big endian). Managing the data in a bi-endian configuration is the responsibility of the application.