Software Development Kit for Multicore Acceleration Version 3.1

		,	
	1	V	
		_	
		-	
		٧	

IDE Tutorial and User's Guide

Software Development Kit for Multicore Acceleration Version 3.1

		,	
	1	V	
		_	
		-	
		٧	

IDE Tutorial and User's Guide

Note

Before using this information and the product it supports, read the information in "Notices" on page 153.

Edition notice

This edition applies to version 3, release 1, modification 0, of the IBM Software Development Kit for Multicore Acceleration (Product number 5724-S84) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2006, 2008. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this publication	. V . v
Part 1. Getting started	1
Chapter 1. Overview of the IBM SDK for Multicore Acceleration Integrated Development Environment.	. 3
Chapter 2. Supported operating environments	. 5
Chapter 3. Installing IDE. Troubleshooting: 32–bit IES Eclipse fails to load on 64–bit Fedora 9	. 7 . 9 . 10
Chapter 4. Uninstalling IDE	11
Part 2. Using IDE	13
Chapter 5. Creating a Cell/B.E. environment	15
Chapter 6. Running IDE on Cell/B.E.	17
Chapter 7. Running IDE on Power-PC systems	21
Chapter 8. Using the CDT XLC error parser	23
Chapter 9. Configuring public key-based authentication using OpenSSH	25
Chapter 10. Troubleshooting and work-arounds	27
Part 3. IDE tutorial	29
Chapter 11. IDE tutorial. . Getting started - open Eclipse . Getting started - select C/C++ Perspective . SPU project - create SPU project . SPU project - select the C Project Wizard .	31 33 34 35 36

SPU project - enter the project name and define the
project type
SPU project - configure SPU Project
SPU project - add directory the SPU compiler
include paths list
SPU project - add the directory path
SPU project - create a new source file 41
SPU project - enter the new source file name 42
SPU project - effet the few source file
SPU project - earl the source me
DDL and the DDL and the project
PPU project - create a PPU executable project 45
PPU project - create the PPU project
PPU project - reference the SPU project - step 1 47
PPU project - reference the SPU project - step 2 48
PPU project - finish creating the PPU project 49
PPU project - configure the PPU project 50
PPU project - select the C/C++ build options 51
PPU project - view the Manage Configurations
window
PPU project - add the libspe2 library to the library
linker list - step 1 53
PPI I project - add the libera? library to the library
linkor list stop 2
DDL project add the CDL eventtable as an embed
CDL investment of the 1
SPU input - step 1
PPU project - add the SPU executable as an embed
SPU input - step 2
DDU project add the CDU executeble as an embed
rro project - add the Sro executable as an embed
SPU input - step 3
SPU input - step 3 57 PPU project - configure additional settings 58 PPU project - create another new source file 59 PPU project - enter the name for the new source file 59 PPU project - edit the source code file 60 PPU project - edit the source code file 61 Cell/B.E. simulator - create the local Cell/B.E. 63 Cell/B.E. simulator - configure the simulator 64
SPU input - step 3 .
SPU input - step 3 .
SPU input - step 3 .
SPU input - step 3 .
SPU input - step 3 .
SPU input - step 3 .
SPU input - step 3

Debug the application - switch to the $C/C++$	
perspective	77
Dynamic profiling tool	78
Dynamic profiling tool - change SPU modes in the	
simulator window	79
Dynamic profiling tool - change the SPEs to pipeline	
mode	80
Dynamic profiling tool - launch the PPU application	81
Dynamic profiling tool - view the results of the	
dynamic performance analysis	82
SPU timing tool - static timing analysis	83
SPU timing tool - view the SPU timing output	84
Simulator console - toggle the simulator console	01
view	85
Simulator consolo soloct the simulator TCL consolo	86
Simulator console - select the simulator TCL console	00
simulator console - view the simulator's TCL	07
Console	07
simulator console - use the simulator's Linux	00
	00
ALF API wizard.	89
ALF API wizard - start the ALF IDE wizard	90
ALF API wizard - enter a project name	91
ALF API wizard - change general parameters	92
ALF API wizard - add ALF buffers	93
ALF API wizard - add the first buffer	94
ALF API wizard - add a second buffer	95
ALF API wizard - add the third buffer	96
ALF API wizard - finish using the ALF IDE wizard	97
ALF API wizard - ALF IDE wizard complete	98
ALF API wizard - Create new launch configuration	99
ALF API wizard - add a new upload rule	100
ALF API wizard - create a new upload rule - step 2	101
ALF API wizard - select the shared library .so file	102
ALF API wizard - launch an ALF application	103
IDE PDT plugin	104
IDE PDT plugin - set up the C Linker flag	105
IDE PDT plugin - set up the C compiler flag	106
IDE PDT plugin - rebuild the project with PDT	
flags	107
IDE PDT plugin - create a PDT configuration file	108
IDE PDT plugin - open the wizard	109
IDE PDT plugin - enter a file name	110
IDE PDT plugin - select event groups	111
IDE PDT plugin - select sub-events	112
IDE PDT plugin - set colors for events	113
IDE PDT plugin - modify the XML configuration	
file	114
IDE PDT plugin - using the Profile Launcher	115
IDE PDT plugin - launch the application with PDT	
support	116

IDE PDT plugin - check generated trace files
Part 4. ALF for IDE programmer's guide
Chapter 12. ALF for IDE overview 123
Chapter 13. Data distribution overview125Practice with data distribution directives
Chapter 14. ALF IDE wizard overview 131
Chapter 15. Programming tips using the ALF IDE wizard
Chapter 16. Programming with ALFIDE wizard.135Using the ALF IDE wizard to create an ALF project136ALF IDE wizard parameters137Using user-defined types140Example: Matrix addition140
Chapter 17. Platform-specific constraints for ALF IDE Wizard on Cell/B.E. architecture
Part 5. Appendixes 147
Appendix A. Related documentation 149
Appendix B. Accessibility features 151
Notices 153 Trademarks 155 Terms and conditions 155
Index

About this publication

This publication describes in detail how to use the IBM Software Development Kit for Multicore Acceleration Integrated Development Environment for Cell Broadband Engine[™] (IDE).

Who should use this book

The target audience for this document is application programmers using the IBM Software Development Kit for Multicore Acceleration (SDK). You are expected to have a basic understanding of programming on the Cell Broadband Engine (Cell/B.E.) platform and common terminology used with the Cell/B.E. platform.

Typographical conventions

The following table explains the typographical conventions used in this document.

Typeface	Indicates	Example
Bold	Lowercase commands, library functions.	void sscal_spu (float *sx, float sa, int n)
Italics	Parameters or variables whose actual names or values are to be supplied by the user. Italics are also used to introduce new terms.	The following example shows how a test program, <i>test_name</i> can be run
Monospace	Examples of program code or command strings.	int main()

Table 1. Typographical conventions

Related information

For a list of SDK documentation, see Related documentation.

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this publication, send your comments using IBM Resource LinkTM at http://www.ibm.com/servers/ resourcelink. Click **Feedback** on the navigation pane. Be sure to include the name of the book, the form number of the book, and the specific location of the text you are commenting on (for example, a page number or table number).

Part 1. Getting started

These topics describe what IDE is, the hardware requirements and how to install IDE.

- Chapter 1, "Overview of the IBM SDK for Multicore Acceleration Integrated Development Environment," on page 3
- Chapter 2, "Supported operating environments," on page 5
- Chapter 3, "Installing IDE," on page 7
- Chapter 4, "Uninstalling IDE," on page 11

Chapter 1. Overview of the IBM SDK for Multicore Acceleration Integrated Development Environment

The IBM[®] SDK for Multicore Acceleration Integrated Development Environment (IDE) is built upon the Eclipse and C Development Tools (CDT) platform.

It integrates the Cell/B.E. GNU tool chain, compilers, IBM Full-System Simulator (simulator) for the Cell/B.E. processor, and other development components to provide a comprehensive, user-friendly development platform that simplifies Cell/B.E. application development. The key features include the following:

- A C/C++ editor that supports syntax highlighting, a customizable template, and an outline window view for procedures, variables, declarations, and functions that appear in the source code
- A rich visual interface for the Power Processing Element (PPE), Synergistic Processing Element (SPE), and the GNU debugger (GDB)
- · A seamless integration of simulator into Eclipse
- An automatic builder, performance tools, and several other enhancements

Chapter 2. Supported operating environments

The following are the prerequisites for using IDE for all operating environments.

Architecture: x86, x86_64, PPC, PPC_64, Cell/B.E.

Operating System: Linux[®]

Software:

- Cell BE SDK 3.1
- Java[™] 1.5 (IBM or Sun.)

Note: The Java^{TM} that comes installed in some Linux[®] installations (GCJ) is *not* sufficient.

Chapter 3. Installing IDE

This topic describes how to install IDE.

If you have installed the SDK and wants to use the optional IES Eclipse IDE component, install IDE by following these steps:

- The GCJ JRE installed with Linux distributions is *not* sufficient for IDE. Download and install J2SE Version 5.0, SR8 or greater for 32-bit xSeries[®] from http://www.ibm.com/developerworks/java/jdk/linux/download.html. SDK provides an IBM 32-bit JRE 1.5 SR8 that you can install by invoking the following commands as root (admin):
 - yum install ibm-java2-ppc-jre (PowerPc version)
 - yum install ibm-java2-i386-jre (i386 version)
- 2. Configure your environment to use the new JRE.
 - a. Eclipse uses the Java virtual machine (JVM) to which your Linux **PATH** variable points. To find out which version of Java is being used, run the command:

java -version

If this returns a JVM other and GCJ, you can go to next step.

b. When you start Eclipse, use the -vm command to specify the IBM JVM. For example:

eclipse -vm /opt/java/jdk1.5.0_08/jre/bin

- c. If you want to modify your environment permanently, configure your PATH variable to reference the IBM JVM. You can modify your PATH variable by editing the file \$HOME/.bash_profile. For example, add this line: PATH=\$HOME/bin:/opt/java/jdk1.5.0_08/jre/bin:\$PATH
- **3**. Install the IDE rpm. To install the IDE rpm, log in as root (admin) user and perform one of the two installation methods:
 - Download the rpm from the developerWorks[®] IDE site. To install it, use the rpm command:
 - > rpm -ivh cellide-3.1.0-[RELEASE_NUM].[ARCHITECTURE].rpm
 - If you have the cell-install rpm installed, you can use yum as follows:
 yum install cellide

This procedure installs an IES Eclipse bundled with CDT 4.0, PTP Remotetools 2.0 and the latest IDE plug-ins into the /opt/cell/ide/eclipse directory.

- 4. Install the Accelerated Library Framework (ALF) IDE template, Performance Debugging Tool (PDT) and SPU Timing packages.
 - a. If you want to use the ALF IDE Wizard, you need to install the ALF IDE template rpm (alf-ide-template). Make sure you have the SDK's cell-install rpm package installed, then invoke the following command:
 - > yum install alf-ide-template
 - b. To enable the PDT feature on IDE, make sure you have the SDK's cell-install rpm package installed, then invoke the following command: yum install pdt
 - **c.** To use the SPU Timing feature, make sure you have the SDK's cell-install rpm package installed, then invoke the following command:

yum install cell-spu-timing

What to do next: For more information about IDE, see the Eclipse help topic.

To access IDE help, in Eclipse, click: **Help → Help Contents → IDE for Cell Broadband Engine SDK**.

Troubleshooting: 32-bit IES Eclipse fails to load on 64-bit Fedora 9

This occurs because there are no specific 32–bit libraries provided with the Fedora 9 x86_64 version.

Solution:

- 1. Install the cellide package.
- Install libcairo 32-bit version: yum install libcairo.i386
- Install libXtst 32-bit version: yum install libXtst.i386

Troubleshooting: 32-bit IES Eclipse fails to load on x86 with RHEL 5.2

This is a known issue caused by a library called xulrunner. The original installation of this library in RHEL 5.2 makes 32-bit IES Eclipse to crash. To solve this problem, remove the original version of xulrunner from your RHEL 5.2 installation: yum erase xulrunner

Chapter 4. Uninstalling IDE

This topic describes how to uninstall IDE for all platforms.

To uninstall IDE, you must remove the cellide package from the system. You can do this in many ways, for example:

- As root user, issue the command: yum erase cellide
- Use any graphical packaging management tool to remove the cellide package

When you remove the cellide package, the following components are deleted:

- IES Eclipse bundled with CDT 4.0 and PTP Remotetools 2.0
- The IDE plug-ins installed in the /opt/cell/ide/eclipse directory

Note: The IDE uninstallation process does not erase any plug-ins or workspaces located on the user's space, but it does erase all the contents of /opt/cell/ide/eclipse and its subdirectories.

Part 2. Using IDE

These topics describe how to use IDE to run and debug applications.

It describes the following topics:

- Chapter 5, "Creating a Cell/B.E. environment," on page 15
- Chapter 6, "Running IDE on Cell/B.E. systems," on page 17
- Chapter 7, "Running IDE on Power-PC systems," on page 21
- Chapter 8, "Using the CDT XLC error parser," on page 23
- Chapter 9, "Configuring public key-based authentication using OpenSSH," on page 25
- Chapter 10, "Troubleshooting and work-arounds," on page 27

Chapter 5. Creating a Cell/B.E. environment

This topic describes the Cell/B.E. environments in which you can run IDE.

You can use IDE to run and debug Cell/B.E. applications in any of the following types of Cell/B.E. environments:

- Attached Cell/B.E. simulator Cell/B.E. simulator running on a remote system. You must manually start the simulator on the remote system. This is different to the **remote Cell/B.E. simulator** environment, where IDE starts the simulator on the remote host.
- **Cell/B.E. box** Physical Cell/B.E. system (remote or native Cell/B.E. system running IDE).
- Local Cell/B.E. simulator Cell/B.E. simulator running on the native system.
- **Remote Cell/B.E. simulator** Cell/B.E. simulator running on a remote system. IDE starts the remote simulator. This is different to the **attached Cell/B.E. simulator** environment, whereby you must start the simulator manually on the remote host.

Before you can run or debug a Cell/B.E. application, you must create and start a Cell/B.E. environment as following:

- 1. Open the Cell Environments view.
- 2. Right click any of the four available Cell/B.E. machine types, then select **Create**.
- 3. Enter any necessary configuration values, then click Finish.
- 4. Start the environment. To do this, click the green **Start** arrow button. After the Cell/B.E. environment has started, it is ready to be used for Cell/B.E. application deployment.

More information:

- Running IDE on Cell/B.E. systems
- Running IDE on Power-PC systems

Chapter 6. Running IDE on Cell/B.E. systems

IDE can be run on a system with a Cell/B.E. processor (as of version 3.0).

The following topics describe how to run IDE on such a system:

- "Managing builder paths for Cell/B.E. systems"
- "Debugging Cell/B.E. applications"
- "Running Cell/B.E. applications" on page 19

Managing builder paths for Cell/B.E. systems

The GNU and XL tool paths are at different locations on a native Cell/B.E. system than on a "typical", non-Cell/B.E. system. IDE's default paths reflect the default Cell/B.E. SDK tool paths of a non-Cell/B.E. computer, so these paths must be corrected. This can be done as follows:

- 1. After installing IDE, open Eclipse preferences. To do this, click **Window** → **Preferences**.
- 2. Open the GNU Tools Managed Build Paths preference page: Cell → Managed Builder Paths → GNU Tools Managed Build Paths.
- **3**. Click **Find it!** to automatically find the GNU tools location, or click **Browse** to manually select the location
- 4. Do the same for the XL Tools Managed Build Paths preference page

Debugging Cell/B.E. applications

Debugging Cell/B.E. applications while running the IDE on a native Cell/B.E. machine requires a few changes to the debugger configuration in order to function properly. Unlike attempting to debug Cell/B.E. applications remotely via SSH, debugging locally does not require the use of gdbserver. Also, instead of creating a C/C++ Cell Target Application debug configuration, you can simply create a C/C++ Local Application configuration (see Figure 1 on page 18).

Note: Other debugger configurations and combinations are possible, only one of them is described here.

- 1. Click **Run > Debug...** to open the debug configuration window.
- 2. Create a new C/C++ Local Application debug configuration (Figure 1 on page 18).

Perilter text C/C++ Attach to Local Application C/C++ Cell Target Application C/C++ Local Application C/C++ Local Application C/C++ Local Application C/C++ Local Application C/C++ Destmortem debt Destmore Des	Configure launch settings from this dialog: Press the 'New' button to create a configuration of the selected type. Press the 'Duplicate' button to copy the selected configuration. Press the 'Delete' button to remove the selected configuration. Press the 'Filter' button to configure filtering options. Edit or view an existing configuration by selecting it. Configure launch perspective settings from the <u>Perspectives</u> preference page.
SPU Timing (Standard Make Projects)	

Figure 1. Creating a new C/C++ Local Application

After you have created a new **C/C++ Local Application** launcher, you must properly configure the **Debugger** tab (see Figure 2 on page 19):

- 1. Select the **gdb/mi** debugger.
- 2. Change the **GDB Debugger** to **ppu-gdb** in order to be able to debug the SPU parts of the application as well as the PPU part.
- 3. Change the GDB command set field to Standard (Cell BE).

î 🗎 🗙 🕒 ⊅×	Name: simple	
ype filter text C/C++ Attach to Local Application C/C++ Cell Target Application C/C++ Local Application C/C++ Local Application C/C++ Postmortem debugger C/C++ Postmortem debugger C/C++ Postmortem debugger C/C++ Postmortem debugger C/C++ Postmortem debugger C/C++ Cell Target Application C/C++ Cell Target Application Junt Junt Junt Plug-in Test PRemote Java Application SPU Timing (Standard Make Projects) SWT Application	 Main & Arguments & Environment Debugger Debugger: gdb/mi Stop on startup at: main Advanced. Debugger Options Main Shared Libraries GDB debugger: ppu-gdb GDB command file: .gdbinit (Warning: Some commands in this file may interfedebugger, for example "run".) GDB command set: Standard (Cell BE) Protocol: mi Verbose console mode 	Browse Browse Browse Browse

Figure 2. Correctly configured Debugger tab

Running Cell/B.E. applications

To run applications locally on a Cell/B.E. system, you can use either the C/C++ Local Application or the C/C++ Cell Target Application launch configuration.

The C/C++ Cell Target Application launcher was initially designed to allow for the running/debugging of applications on a Cell/B.E. simulator or system, while the development of that application was done on a computer without a Cell/B.E. processor. However, you can use this same launcherto deploy an application locally if you are running IDE on a system with a Cell/B.E. processor. In order to accomplish this, you need to create a Cell Box environment (in the Cell Environments view) with the configuration option Localhost selected (instead of Remote host being selected). After you have clicked the green arrow to start the localhost Cell Box, it can then be used as the target environment in the C/C++ Cell Target Application launcher.

Chapter 7. Running IDE on Power-PC systems

IDE can be run on a Power-PC (PPC) system.

The following topics are worth discussing if you plan to use this platform to run the IDE:

• "Managing builder paths for a PPC system"

This section discusses any issues that may come up when running the Cell IDE on a Power-PC (PPC) system.

Managing builder paths for a PPC system

The SDK's GNU and XL tool paths are at different locations on a PPC-based system than on systems with Intel[®] processors. IDE's default paths reflect the default Cell/B.E. SDK tool paths of a Intel-based computer, so these paths must be corrected if you wish to run IDE on PPC. This can be done as follows:

- 1. After installing IDE, open Eclipse preferences. To do this, click **Window** → **Preferences**.
- 2. Open the GNU Tools Managed Build Paths preference page: Cell → Managed Builder Paths → GNU Tools Managed Build Paths.
- **3**. Click **Find it!** to automatically find the GNU tools location, or click **Browse** to manually select the location
- 4. Do the same for the XL Tools Managed Build Paths preference page

Chapter 8. Using the CDT XLC error parser

This topic describes how to use the CDT XLC error parser with IDE.

The IDE is integrated with the Cell/B.E. GNU compilers and also with the Cell BE XL compilers. The error messages printed by these compilers are parsed by the appropriate error message parser in order to identify the error information and display it correctly in the graphical user interface. This makes it easier for you to solve problems that the compilers encountered with your source code.

The GNU C Compiler error message parser comes integrated with CDT. Previous versions of the CDT did not support XL C/C++ compiler error messages and for this reason IDE provides its own XLC error parser. However, from version 3.1.2 onwards CDT provides an XL C/C++ error parser, which you can install from the CDT update site. If you prefer to use this XL C/C++ error parser instead of the one provided by IDE, you need to follow the following instructions to the install CDT XLC error parser as it is not installed by default when CDT is installed:

1. Install Java, Eclipse and CDT as described in steps 1 through 4 of the IBM SDK for Multicore Acceleration IDE Installation Instructions.

Note: The CDT XLC Error Parser is available from **CDT version 3.1.2** onwards.

- 2. Start Eclipse. Make sure you start Eclipse with a user that has write permission to the Eclipse installation directory.
- 3. From the menu, select Help > Software Updates > Find and Install...
- 4. Select Search for new feature to install, then press Next.
- 5. Click New Remote Site
- 6. Enter a name in the **Name** field that identifies the update site, for example, **CDT Update Site**.
- Enter the following into the URL field http://download.eclipse.org/tools/cdt/releases/callisto
- 8. Click **OK**. The recently created entry appears selected in the **Sites to include in search** field.
- 9. Click Finish and follow the on-screen instructions.

After you have completed the above steps, a **CDT XLC Error Parser** entry appears in the **Error Parser** tab. You can access this tab through any of the following:

- Select Window → Preferences... → C/C++ → Make → New Make Projects
- From the C/C++ Build dialog page of a Managed Make C/C++ project Properties
- From the C/C++ Make Project dialog page of a Standard Make C/C++ project Properties
- From the final dialog of the project creation wizard for Managed Make C/C++ and Standard Make C/C++ projects.

From this **Error Parser** tab, you can deselect the IDE **XLC Error Parser** and select the **CDT XLC Error Parser**.

Chapter 9. Configuring public key-based authentication using OpenSSH

The following topics describe how to configure your client and server (as well as how to configure the IDE's environments) to use the public key authentication method.

IDE establishes connections to remote Cell/B.E. environments using Secure Shell (SSH). Two SSH authentication methods are supported:

- Password-based
- · Public key-based

Step 1. Verify the software

First, confirm that OpenSSH is the SSH software installed on the client system. Public key generation can be different for different implementations of SSH. Use the ssh-V command to check the version of your SSH software as follows:

> \$ ssh -V OpenSSH_4.3p2, OpenSSL 0.9.8b 04 May 2006

Step 2. Generate a key pair

You must generate a RSA public/private key pair on the client system. The public key is copied to the remote server that is being connected to, the private key remains on the client system in a secure location. Use the ssh-keygen command to generate the key pair as follows:

```
client$ mkdir ~/.ssh
client$ chmod 700 ~/.ssh
client$ ssh-keygen -q -f ~/.ssh/id_rsa -t rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Step 3. Distribute the key

Next, copy the public key ~/.ssh/id_rsa.pub to the remote server. After the public key has been copied, you need to append the key's contents to the file ~/.ssh/authorized_keys:

```
client$ scp ~/.ssh/id_rsa.pub user@server.example.com:
client$ ssh user@server.example.com
```

```
server$ mkdir ~/.ssh
server$ chmod 700 ~/.ssh
server$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
server$ chmod 600 ~/.ssh/authorized_keys
server$ rm ~/id_rsa.pub
```

Step 4. Confirm the functionality

Attempt to connect to the server to confirm that the public key authentication method is working. If you are using the public key authentication method, you are asked for the passphrase when you connect:

```
client$ ssh -o PreferredAuthentications=publickey
user@server.example.edu
Enter passphrase for key '/home/user/.ssh/id rsa':
```

Step 5. Configure the Cell/B.E. environment

Finally, you need to configure the Cell/B.E. environment inside the IDE.

Note: Public key authentication can be used on any of the Cell/B.E. target environment types except for the **Local Cell Simulator** type.

The following steps explain how to configure your Cell/B.E. Environment:

- 1. In Eclipse, open the **Cell Environments** view.
- 2. Open the cell environment's configuration window by either creating a new environment (right click environment type and select **Create**) or by editing an existing environment (right click the existing environment and select **Edit**).
- 3. Select the option Public key based authentication.
- 4. Press the **Browse** button next to the field **File with private key**, and select the private key file ~/.ssh/id_rsa.
- 5. Enter the passphrase that you entered when you created the public/private key pair.
- 6. Click Finish and use the green start arrow to start the environment.

Chapter 10. Troubleshooting and work-arounds

This topic describes solutions to problems you can encounter when you use IDE.

The following are known issues with IDE:

- "Using the Java API with the simulator"
- "Reconnecting to the GDB server"
- "Ensuring that the remote launch directory is writable for all users"
- "Configuring the localhost" on page 28
- "Specifying MAC addresses for the local Cell/B.E. simulator" on page 28

Using the Java API with the simulator

There is a known bug with the Cell/B.E. simulator's capacity to use the Java API along with the simulator GUI window. To enable the use of the Java API with the simulator, you must set the option **USE_JAVA_API** to true: USE_JAVA_API=true

This parameter can be found in the file **parameters.properties**, which is located at: eclipse/plugins/com.ibm.celldt.simulator/com/ibm/celldt/simulator/conf

If the **USE_JAVA_API** option is set to true, the simulator GUI window cannot be used.

Reconnecting to the GDB server

If you are debugging a Cell/B.E. application remotely using gdbserver and the connection to the gdbserver is broken, IDE tries to reconnect automatically. However, this is not successful even though the gdbserver is still running on the target system.

If you stop this debug session and try to start a new one, IDE issues an error because the gdbserver is still occupying the previous configuation port on the target system and IDE is not able to launch another instance of gdbserver in the same port unless the debug launcher configuration is modified.

The workaround for this is to kill the gdbserver application on the target system before you launch another debug session again.

Ensuring that the remote launch directory is writable for all users

The Cell/B.E. Target Application launcher automatically creates a remote directory for launching the application. By default, this directory is given by following macros:

\${system_workspace}/\${user_workspace}/\${project_name}/\${timestamp}

If you are sharing the remote (Cell/B.E.) system, make sure that all users can write to the directory provided by \${system_workspace}.

The first user that connects to the machine creates \${system_workspace}, subsequent users create their directories inside \${system_workspace}. However, if \${system_workspace} was not created with full write privileges, then the subsequent users will not be able to launch.

Configuring the localhost

When IDE uses the Cell/B.E. Target Application launcher and Local Cell/B.E. Simulator, it calls Inet4Address.getLocalHost() several times. In some cases, this method can fail if your system is not configured correctly. This failure can result in IDE refusing to create new simulator environments or cause failures or both with the Cell/B.E. Target Application launcher.

If you encounter this problem, check the /etc/hosts file and ensure that the following line is present:

127.0.0.1 localhost mymachine

where mymachine is the name of the system as given by the environment variable \$HOSTNAME.

Specifying MAC addresses for the local Cell/B.E. simulator

The local simulator configuration window allows you to choose your own MAC address. If you decide to specify your own MAC address, the following rules apply:

- The first number of the address MUST NOT be odd. It MUST be even. It should be n*4+2 (02, 06, 10,)
 - If the first number is odd, then SSH will not connect, even if simulator pings successfully
 - If the first digit is not n*4+2, then it does not conform to RFC
- It is recommended that the second and third numbers are 00
Part 3. IDE tutorial

Chapter 11. IDE tutorial

This introductory tutorial explores IDE and provides click-by-click lessons on how to use many of the main features of the IDE.

Purpose of this tutorial

In this tutorial you will learn how to:

- · Create, build, and run PPU/SPU managed make projects
- Use the local Cell/B.E. simulator environment
- Use the performance analysis tools (static and dynamic)
- Configure and use the C/C++ Cell Target Application launcher to run and debug your Cell/B.E. applications
- Use the ALF IDE wizard by means of an example usage scenario
- Use the IDE PDT to generate trace profiles

Preparation

Before you start with the tutorial you must prepare your system. You must have IDE installed and running on the computer you want to use for the tutorial. For information about how to install IDE, refer to Chapter 3, "Installing IDE," on page 7.

Learning time

You should allow two and a half hours to complete the tutorial.

Conventions

The following conventions are used in the task descriptions:

- Select File → New means "Select item 'New' from the 'File' menu."
- Click **OK** means "click the OK button".

Table of contents

Click one of the following sections to jump to a particular section of the tutorial, or click here to start at the beginning.

You can return to the table of contents at any time by clicking the "Parent topic" link in the tutorial.

- 1. "SPU project create SPU project" on page 35
- 2. "PPU project create the PPU project" on page 46
- 3. "Cell/B.E. simulator create the local Cell/B.E. simulator" on page 63
- 4. "Create an application launch configuration" on page 68
- 5. "Debug the application" on page 76
- 6. Dynamic/static performance analysis
- 7. "Simulator console toggle the simulator console view" on page 85
- 8. Accelerated Library Framework API Wizard
- 9. "View the Eclipse preferences window" on page 118

- 10. "View the IDE environment preferences" on page 119
- **11**. "IDE PDT plugin" on page 104

Getting started - open Eclipse

With CDT and IDE installed into your eclipse directory, open Eclipse and switch to the C/C++ perspective. To do this click **Window → Open Perspective → Other...**

3		Java - Eclipse SDK			
Eile Edit Source Refactor	Navigate Search Project Bun	Window Help			
th• 🗟 🗟 \$• 0• 9• &	🖶 G- 🕭 🛷 (0)- (0)- (0)	New Window			🗊 🖏 Java
🖩 Packa 😫 🕆 Hierar 😁 🗆		Open Perspective	S Debug	- 0	😹 Outline 😫 🔍 🗖
00000000		Show View	📲 Java		An outline is not
		Customize Perspective Save Perspective <u>A</u> s <u>Reset Perspective</u> <u>Close Perspective</u> Close All Perspectives	V Java Browsing Other		avanaute.
		Navigation	>		
		₩Orking Sets Preferences			
	E Problems ≅ e Javadoc ⊛ D 0 errors, 0 warnings, 0 infos	eclaration			3 × °C
	Description R	esourc Path Lo	cation		

Getting started - select C/C++ Perspective

Select C/C++, then click OK.

D	B # G. B. A. B. B. B. C. C. C.		🖽 🖏 Java
🛿 Packa 😆 👔 Hierar 🍧	3	• 0	🗄 Outline 😫 🦳 🗖
	3 2 2 2 2 3 2 3 2 3 3 3 2 3 3 2 3 3 2 3	Open Parspective Image: Ce++ VS Repository Exploring ebug wa (default) wa Browsing wa Type Hierarchy lug-in Development esource esam Synchronizing OK Cancel	An outline is not available.
	🗶 Problems 😫 🤍 Avadoc 🗟 Declarat	ion	» × • □
	0 errors, 0 warnings, 0 infos		

SPU project - create SPU project

Objectives: After you have completed this part of the tutorial you will know how to:

- Create and configure an SPU project
- Add the directory that contains the profile.h header file to the list of the compiler's include paths so that you can use the dynamic performance analysis tool
- Create a C source file
- Automatically build the project

Create a new C project. Click **File > New > Project**.

\$		C/G++ - Eclipse/SDK		
Elle Edit Refactor Navigate S	earch Project Bu	n <u>Window H</u> elp		
New Open file	Shift+Ait+N →	C Project		E [™] C/C++ **
open File	1997 C 199	E C++ Project	- 0	🖹 Outline 🕴 🖲 Make Ta 📮 🗖
Close All	CERI+W	Convert to a C/C L L Make Project		An outline is not available.
Close An	Shirt+Ctri+W	Source Folder		
<u>Save</u>	Ctri+5	Source rolder		
Save All	Shift+Ctrl+S	Source File		
Revert	Sunct Sull's	😵 Header File		
Mova		🗈 File		
Bename	F2	@ Class		
Refresh	F5	📬 Example		
Convert Line Delimiters To	>	C Qther		
Erint	Ctrl+P			
Switch Workspace	>			
in Import in Export				
Properties	Alt+Enter			
Exit		1		
	(t. Proble 0 errors, Descript	ms 2 Tasks Console Properties 0 warnings, 0 infos ion Resourc Path	B Remote Environments	\$ × •0

SPU project - select the C Project Wizard

The **New Project** wizard is displayed.

- 1. Expand the section **C**.
- 2. Select C Project.

S File Edit Defector Nevineta Conc	h Designt Due Winder	C/C++ - Eclipse SDK		[_][e][X
The Full Relactor Mavigate Sean	n Froject Bun Window	v <u>H</u> eip In a lair àir to an an		F DC/C++ *
Project Explorer 🛛 🕞 🇞 🔻	Select a wizard			Cutline is not available.
	Wizards:	4d		
	Lype micr text [#] Java Project [#] Java Project from Ex [#] Plug-in Project ^b © General ^b △ ALF for Cell ^a ⊂ C ^c ⊂ C+	Existing Ant Buildfile		
	Problems # C	Tasks Console Properties	Remote Environments	
	0 errors, 0 warning Description	s, 0 infos Resourc Path	Location	
n.			1	

3. Click Next.

SPU project - enter the project name and define the project type

For **Project name**, enter SPU. You can select the appropriate **Project Type** (Cell PPU Executable, Cell SPU Static Library, and so on) for the project you are creating. When you select a project type, you see the default **Configurations** available for that project type. Later in the tutorial you will learn how to create new build configurations, as well as how to modify the existing built-in configurations' settings.

- 1. For Project Type, select Cell SPU Executable.
- 2. Click Finish.

3	a		= 0 (a
File Edit Refactor Navigate Search	C Project Create C project of selected type	*	≅ ℝC/C++ *
🕒 Project Explorer 🛪 🛛 😑 🐄 🥗		- (🛛 🕃 Outline 🕱 💿 Make Ta 🔍 🗆
	Project name: SPU © Use default location Location: [/home/danielhb/workspace_testing/SPU Browse		An outline is not available.
	Project types: Toolchain:		
	> Executable SPU GNU Debug Tool Chain > Shared Library Static Library > Static Library SPU GNU Release Tool Chain > Static Library(XL C/C++) Shared Library(XL C/C++) > Shared Library (XL C/C++) Cell PPU Static Library > Cell PPU Static Library Cell SPU Shared Library > Cell SPU Static Library Cell SPU Static Library > Cell SPU Static Library Cell SPU Static Library		
	$\ensuremath{\varnothing}$ Show project types and toolchains only if they are supported on the platform	F	» × = □
0*	© < Back Next > Finish Cancel		

SPU project - configure SPU Project

In the C/C++ Projects view:

- 1. Right click the SPU project.
- 2. Select Properties.

		C/C++ - Eclipse SDK			
Edit Refactor Navigate Search	Project Bun Window	Help			
- 🔄 📑 🗋 💼 - 🚳 - 🗟 - 🞯 - 🗍 🗞 - 1	8- \$- Q- Q- @	4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			E C/C++
roject Explorer 🗈 💿 😓 👻 🖷 t)			• 0	🐉 Outline 🛛 💿 Make Ta
New	>				An outline is not available.
Open in New Window					
Сору	Ctrl+C				
🗇 Paste	Ctrl+V				
× Delete	Delete				
Moye	000000				
Rename	F2				
im Import					
🖆 Export					
Clean Project					
8 Refresh	F5				
Close Project					
Close Unrelated Projects					
Exclude from build					
Build Configurations	>				
Make targets	>				
Index	>				
Convert To					
Bun As	>	orke D Concolo - Proportion	B. Romoto Environmente		4. V
Debug As		O infes	B Remote Environments		4
Profile As	> warnings	Pacoure Path	Location		
Team	> 201	Resource Facili	Location		
Compare With	>				
Restore from Local History					
FDE IOUIS					
Properties	Art+Enter				

SPU project - add directory the SPU compiler include paths list

You need to add the directory that contains the **profile.h** header file to the list of the compiler's include paths so that you can use the dynamic performance analysis tool.

- 1. Click C/C++ Build in the left pane.
- 2. Under SPU GNU C Compiler with Debug Options, select Directories.
- 3. In the **Include Paths** pane on the right, click the **Add** button.

3	Properties f	stik or SPU S	
type filter text	Settings	¢• ⇔-	C/C++ *
Resource Builders	Configuration: spu-gnu-debug	Manage configurations	ake Ta □□ ailable.
Discovery optic	Tool settings Build steps Build artifact Bin	nary parsers o Error parsers	
Environment Settings Tool chain edito Variables XL C/C++ Comj ▷ C/C++ General Project Reference Refactoring Histo Run/Debug Settin	 SPU GNU Assembler General SPU GNU C Linker General Libraries Miscellaneous Shared Library Settings Profile & Trace Overlay SPU GNU C Compiler with Debug Options Preprocessor Symbols Directories Optimization Debugging Warnings Miscellaneous 	Include paths (-i)	3 7 90
		Restore Defaults Apply	

SPU project - add the directory path

- 1. Enter /opt/ibm/systemsim-cell/include/callthru/spu
- 2. Click OK twice to return to the C/C++ Perspective.

S File Edit	•	C/C++ - Eclipse SDK Properties for SPU X	
	type filter text	Settings 🔶 🔶	C/C++ **
I <mark>&</mark> Project I► © SPU	Resource Builders ▽ C/C++ Build	Configuration: [spu-gnu-debug 0] Manage configurations]	ake Ta 😁 🗆 ailable.
	Discovery optic	Tool settings Auild steps Build artifact Binary parsers GError parsers	
	Environment Settings Tool chain edito		
	Variables XL C/C++ Com	S Add directory path Directory:	
	C/C++ General Project Reference Refactoring Histo Run/Debug Settin	[/opt/ibm/systemsim-cell/include/callthru/spu	
		V 8 S OK Cancel Workspace File system	
		🖌 Directories	
		Optimization Pehunging	
		Warnings Miscellaneous Profile & Trace	> 0
	(Restore Defaults Apply	Ì
	Ø	OK Cancel	
0° 6	L		_

SPU project - create a new source file

Create a C source file. Click **File → New → Source File**.

a				
File Edit Refactor Navigate	Search Project Bu	n <u>W</u> indow <u>H</u> elp		
New	Shift+Alt+N >	🔤 C Project		PR PRC/C++ *
Open File		C++ Project		
Close	Ctrl+W	📬 Project		E Outline # Make la
C[ose All	Shift+Ctrl+W	🖻 Convert to a C/C++ Make Project		An outline is not available.
Save	Ctrl+S	Source Folder		
Save As		😅 Folder		
Sav <u>e</u> All	Shift+Ctrl+5	& Source File		
Revert		Header File		
Mo <u>v</u> e		The Class		
Rename	F2	G Class		
Refresh	F5	Example		
Convert Line Delimiters To	>	C Other		
i Print	Ctr1+P			
Switch Workspace	>			
迦 Import				
🖾 Export				
Properties	Alt+Enter			
Exit		1		
	🗄 Problems 🕢	Tasks 📴 Console 🛛 🗖 Properties 🐁	Remote Environments	🙆 🗞 🖆 💷 😷 " 🗖
	C-Build [SPU]			2
	**** Build of	configuration spu-onu-debug for	project SPIL ****	-
	burtu or	configuration spa gila acoug for	project site	
	Nothing to bu	ild for project SPU		1
CP IS SPU				
J. 4 C C , 5			1	

SPU project - enter the new source file name

- 1. In Source File, enter spu.c.
- 2. Click Finish.

•		C/C++ - Eclipse SDK			- n x
<u>File Edit Refactor Navigate Sear</u>	ch Project Bun Window	Help			
1 C3+ 1 = 1 = 1 = 1 = - 63+ C3+ (3+ (3+))	§* @* \$* 0* 9* @	A			≅ ©C/C++ "
B Project Explorer 🛚 👘 🗖				- 0	B Outline II • Make Ta • D
E 4	0	New Source File		(x)	All outline is not available.
e sru	Create a new source file	e.		C	
	Source Folder: SPU			Browse	
	Source File: source				
	0 C-build (51-0)		Einish	Cancel	0 % d 0- 0- °C
	**** Build of configura Nothing to build for p	ation spu-gnu-debug for pro roject SPU	oject SPU ****		
D* 🐸 SPU					<u>U</u>

SPU project - edit the source file

}

A new editor is displayed so you can enter your source code. Copy and paste the following source code into your editor (you will uncomment the commented lines later):

```
#include <stdio.h>
#include <profile.h>
int main(unsigned long long id)
{
    //prof_clear();
    //prof_start();
    printf("Hello Cell (0x%llx)\n", id);
    //prof_stop();
    return 0;
```



SPU project - automatically build the project

Save the source file (**Control + S**). The project is automatically built.

The build output is displayed in the **Console** view, and new resources, such as binaries and includes, are displayed in the **C/C++ Projects** view.

•		
File Edit Refactor Navigate	Search Project Bun Window Help	
10- 1 0 1 1 0- 6- C- C	- % - 8 - \$ - 0 - 4 - \$ \$ 1 b \$ + \$ + \$ + \$ + \$ - \$ - \$ - \$ - \$ - \$ -	≅ © C/C++ **
Project Explorer 🛪 🗧	ື 🖬 spu.c ສ	🖱 🗖 🐉 Outline 🕸 🖉 🖲 Make Ta 🏻 🗖
♥ SPU ▶ Sinaries ▶ Sincludes ▶ Syu-gnu-debug ▶ Oispuc	<pre> finclude <stdio.h> finclude <stdio.h> finclude <stdio.h> int main(unsigned long long id) {//prof_clear(); //prof_start(); printf("Hello Cell (0x%llx)\n", id); //prof_stop(); return 0; } Creblems @ Tasks © Console = Properties & Remote Environments C evid (Cell) </stdio.h></stdio.h></stdio.h></pre>	i stdio.h u stdio.h u profile.h • main(unsigned long long) • • • • • • • • • • • • • • • • • • •
	<pre>**** Build of configuration spu-gnu-debug for project SPU **** make all Building file:/spu.c Invoking: SPU GNU C Compiler with Debug Options spu-gcc .T/pot/ibm/systemsin-cell/include/callthru/spu -00 -g3 -Wall -c -fn fno.inline .mtune=cell .mtloat=fast .mdouble=fast .Wno-main .march=cell .me WT spu.d* .o*spu.o*/spu.c Building target: SPU Invoking: SPU GNU C Linker spu.gcc .wlN o*SPU** ./spu.o Finished building target; SPU</pre>	essage-length=0 -Winline -Wextra - a32 -MMD -MP -MF*spu.d* -
D*	Writable Smart Insert 12:2	

PPU project - create a PPU executable project

Objectives: After you have completed this part of the tutorial you will know how to:

- Create a PPU project
- Configure the PPU project
- Add the SPU executable as an embed SPU input

Now create a PPU executable project that uses the Embed SPU tool in your SPU project.

Click **File** → **New** → **C Project**.

3						
Elle Edit Refactor Navigate	Search Project Bu	n Window Help				
New	Shift+Alt+N >	🔁 C Project		P1 C/C++ *		
Open File		C++ Project				
Close	Ctrl+W	Project	-0	B Outline # Make la - 0		
C <u>l</u> ose All	Shift+Ctrl+W	Convert to a C/C++ Make Project		12 8 8 0 0		
iii Save	Ctrl+S	as Source Folder		stdio.h		
🗟 Save As		🐸 Folder		profile.h		
Save All	Shift+Ctrl+S	Source File		 main(unsigned long long) 		
Revert		Header File				
Mo <u>v</u> e		Chart Chart				
Rename	F2	G Class				
Re <u>f</u> resh	F5	Example				
Convert Line Delimiters To	>	📑 Other				
m Print	Ctrl+P	Deks D Consola 31 D Proportion B	Pomoto Environmente			
Switch Workspace	>	asks Console & Properties @	Remote Environments	0 18 E 0 13* D		
🖮 Import				0		
🖆 Export		configuration spu-gnu-debug for	project SPU ****			
Properties	Alt+Enter					
1 spu.c [SPU]		:/spu.c				
Exit		GNU C Compiler with Debug Option	ns Ithru/spu -00 -03 -Wall -c -fmessage-ler	ngth=0 .Winline .Wextra .		
	fno-inline -m MT"spu.d" -o" Finished buil Building targ Invoking: SPU spu-gcc -Wl,- Finished buil	itune=cell -mfloat=fast -mdouble= ding:/spu.c et: SPU GNU C Linker N-α°SPU -/spu.o ding target: SPU	fast -Who-main -march=cell -mea32 -MMD -	ĀP .HF*spu.d*		
0 items selected						

PPU project - create the PPU project

Create the PPU project:

- 1. For Project name, enter PPU.
- 2. For Project Type, select Cell PPU Executable
- 3. Click Next.



PPU project - reference the SPU project - step 1

The project **SPU** that you created earlier is listed. Because you will embed the **SPU** project's executable into the **PPU** project, you need to reference it.

Click Advanced settings.

•	3		- 0 X
File Edit Refactor Navigate Searc	Select Configurations Select platforms and configurations you wish to deploy on		≅ ₽C/C++ *
Coproject Explorer 18	Project type: Cell PPU Executable Tool-chains: PPU GNU 32 bit Debug Tool Chain Configurations:		Stdio.h
	Ø ≫ ppu-gnu32-debug	Select all Deselect all	• main(unsigned long long)
	A	dvanced settings	
1 s t h f E 1 s f	Use "Advanced settings" button to edit project's properties. Additional configurations can be added after project creation. Use "Manage configurations" buttons either on toolbar or on property © < Back Next > Einish	y pages.	e-length≕0 -Winline -Wextra - ₩0 -MP -MF*spu.d* -
tr ⊌ SPU			, ,

PPU project - reference the SPU project - step 2

- 1. Go to the **Project References** category.
- 2. Click OK.



PPU project - finish creating the PPU project

Click Finish to create the PPU project.

•	0	×	
Eile Edit Refactor Navigate Search	Select Configurations Select platforms and configurations you wish to deploy on		
► SPU	Project type: Cell PPU Executable Tool-chains: PPU GNU 32 bit Debug Tool Chain Configurations:		u stdio.h u profile.h
	Ø @ ppu-gnu32-debug	Select all Deselect all	main(unsigned long long) main(unsigned long long long) main(unsigned long long long long long long long long
	Use "Advanced settings" button to edit project's properties. Additional configurations can be added after project creation. Use "Manage configurations" buttons either on toolbar or on prop	Advanced settings	∍-length=O -Winline -Wextra - WHD -MP -MF*spu.d* -
t 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	The second secon	h Cancel	

PPU project - configure the PPU project

In the C/C++ Projects view:

- 1. Right click the **PPU** project.
- 2. Select Properties.

a		
File Edit Refactor Navigate Search Project Bur	n <u>W</u> indow <u>H</u> elp	
13+ 📾 📾 📾 - 18+ 18+ 18+ 18+ 18+ 19+	💁 🖉 🛷 🗍 🐨 🔯 🗍 82 - 82 - 😎 0 - 80 -	
🕒 Project Explorer 😫 🧧 🖬 spu.c 😫		" 🗆 <table-cell-rows> Outline 🛛 🛞 Make Ta 👕 🗖</table-cell-rows>
E 😵 ऱ #include <st< td=""><td>tdio.h></td><td>12 12 12 12 1</td></st<>	tdio.h>	12 12 12 12 1
Del New	rofile.h>	u stdio.h
▷ 😂 S Open in <u>N</u> ew Window	igned long long id)	u profile.h
Copy Ctrl+C Paste Ctrl+V Delete Delete Moye Rename F2	ar(); rt(); llo Cell (0x%llx)\n", id); p();	 main(unsigned long long)
ia Import ia Exp <u>o</u> rt		
Clean Project Refresh F5 Close Project Close Unrelated Projects	ssks 🖾 Console 🛪 🖉 Properties 🚯 Remote Environments	a a d - c- = -
Exclude from build Build Configurations		
Convert To Bun As >> Debug As >>		
Profile As >> Team >>		
Compare With A Restore from Local History		
B ^e Properties Alt+Enter		

PPU project - select the C/C++ build options

- 1. In the left pane, select C/C++ Build.
 - In the **Configuration** group at the top, you can change the current configuration (for example. ppu-gnu32-debug, ppu-xl32-debug, and so on) or create your own. In the **Builder settings** tab, you can manipulate builder and makefile configurations.
- 2. Click Manage configurations.

•	3		C/C++ - SPII/snu c - Eclinse SDK Properties for PPU		
	type filter text	C/C++ Build		\$ * 5	
Project Explor	Resource Builders	Configuration: p	pu-gnu32-debug	Manage configuration	s
 ▷ C/C++ Build ▷ C/C++ General Project References Refactoring History Run/Debug Settings 		Builder settings Builder Builder type Ø Use default b Build command:	e Behaviour External builder uild command	¢]	long)
		Makefile generat	eefiles automatically Strand Env. Variable Refs in Makefiles) • • •
			Workspace) File	system] [Variables]	
			Restor	e Defaults Apply	
📬 🐸 PPU	Ø			OK Cancel	

PPU project - view the Manage Configurations window

In this window you can rename, remove, or create new build configurations.

Click **OK** to continue.

Sile Edit Refer	0		C/C++ - SPU/spu Pro	c - Eclinse SDK operties for PPU			×	
	type filter text	C/C++ Build					6 • <-	+ *
Project Explor	Resource Builders	Configuration	: ppu-gnu32-debug			0	Manage configurations	• •
<mark>i> e≥ PPU</mark> > e3 SPU	 C/C++ Build C/C++ General Project References Refactoring History Run/Debug Settings 	■ Builder sett Builder Builder type I Use def Build comr Makefile ge I Genera Build locat Build direc	ings e Behaviour External builder PPU : Mar Configuration Dr ppu-gnu32-debug ppu-gnu32-release Set Active New.	nage configurations escription Status Active Remove OK	(Rename Cancel	Makefiles	Variables)	long)
T* 65 PPU	() = () ()					Restore	Defaults Apply K Cancel	

PPU project - add the libspe2 library to the library linker list - step 1

You use libspe2 in your PPU source code, so you need to add the library "spe2" to the linker's list of libraries.

- 1. In the **Tool Settings** tab, select the **Libraries** option, which is under the **PPU GNU 32 bit C Linker** category.
- 2. Click the Add button in the Libraries pane on the right.

Edit Befact	1.1.3.2	Properties for PPU			
type filte	er text Settings			ф• ф-	+ *
oject Explor Builde V C/C++	rce Configuration: ppu-gnu32-debu	g	0	Manage configurations	• •
2U Disc	overy options Tool settings Build steps B	uild artifact 🗟 Binary parsers 🧿 Error parsers			
Envi Sett Tool Varia XL C ▷ C/C++	ronment ings Chain editor bbles (C++ Compil General → DPU GNU 32 bit Assembler ⇒ General → PPU GNU 32 bit Assembler ⇒ General ⇒ Architecture → DPU GNU 32 bit Assembler ⇒ General → DPU GNU 32 bit Assembler ⇒ DPU	Libraries (-I)		Add	long)
Project	t References oring History	Library search pat	th (-L)		D
Run/De	ebug Settings Clubraries @ Miscellaneous @ Shared Library Settings @ Profile & Trace @ Architecture				
	 PPU GNU 32 bit C Compiler Preprocessor Symbols Directories Optimization Debugging 	with Debug Options Math library (-Im) Pthread library (-Im) libspe2 (-Ispe2)	lpthread)		
13			Restore J	Defaults Apply	j I
			-		

PPU project - add the libspe2 library to the library linker list - step 2

Enter spe2, then click **OK**.

•	C/C++ - SPU/spu c - Eclinse SDK Properties for PPU	K	
type filter text Resource Builders	Settings Configuration: ppu-gnu32-debug	چې د. Manage configurations)	+ *
 C/C++ Build Discovery options Environment Settings Tool chain editor Variables XL C/C++ Compil C/C++ General Project References Refactoring History Run/Debug Settings 	 Tool settings #Build steps #Build artifact #Binary pars PPU GNU 32 bit Assembler General Enter Value Libraries (-1) Spe2 Architecture PPU GNU 32 bit C Compiler with Debug Options Preprocessor Symbols Directories Optimization Debugging 	Sers © Error parsers	long)
() ()		Restore Defaults Apply	
	 ♥ Yype filter text Resource Builders ♥ C/C++ Build Discovery options Environment Settings Tool chain editor Variables XL C/C++ Compil ♥ C/C++ General Project References Refactoring History Run/Debug Settings 	Image: ClC++ - SPU/Result - Friinse SDK Properties for PPU Type filter text Resource Builders C(C++ Suid Discovery options Environment Settings Tool chain editor Variables XL C/C++ General Project References Refactoring History Run/Debug Settings PPU GNU 32 bit C Compiler with Debug Options Preprocessor Symbols Directories Optimization Preprocessor Symbols Directories Optimization	C/C++ - SPU/Enu c - Felines SDK Properties for PPU Yype filter text Resource Builders C/C++ Build Discovery options Environment Settings Tool chain editor Variables XL C/C++ Compil Project References Refactoring History Run/Debug Settings PUE ONU 32 bit C Compiler with Debug Options Preprocessor Objectories Optimization Properties for PPU

PPU project - add the SPU executable as an embed SPU input - step 1

In order for the **PPU** project to embed the **SPU** project's executable, you need to add the **SPU** executable as an embed SPU input.

- 1. In PPU GNU 32 bit Embed SPU, select the Inputs option.
- 2. Click the Add button in the Embed SPU Inputs pane.

Sile Edit Defe	3	C/C++ - SPU/enu c - Eclinee SDK Properties for PPU		
	type filter text	Settings	⇔ < -	.
Project Explo	Resource Builders ▼ C/C++ Build	Configuration: ppu-gnu32-debug	□ ♀ 〕 [Manage configurations]	a = 0 • ~
b. ⊴ PPU Þ ⊴ SPU	 CrC++ Built Discovery options Environment Settings Tool chain editor Variables XL CrC++ Compil CrC++ General Project References Refactoring History Run/Debug Settings 	© Tool settings	Embed SPU Inputs	g long)
			Restore Defaults Apply	
to 🐸 PPL	Ø		OK Cancel	

PPU project - add the SPU executable as an embed SPU input - step 2

Click the **Workspace** button.

	and the second second second second	Cabble as				1000	
C1* 🖬 👝 🔒	type filter text	Settings				\$*Q.	++ *
🍐 Project Explo	Resource Builders ▼ C/C++ Build	Configuration:	ppu-gnu32-debug		0	Manage configurations	a = 0
e PPU	Discovery options	Tool settings	PBuild steps Build artifact 🔒	Binary parsers o Error parsers			
> 🐸 SPU	Environment Settings Tool chain editor	♥ ⊕ PPU GNU 3 @General	32 bit Assembler	Embed SPU Inputs	i	A 0 0 0 D	g long)
	Variables XL C/C++ Compil C/C++ General Project References Refactoring History	File:	Add file p	ath			
	Run/Debug Settings	 Architec PPU GNU 3 Preproce Symbols Director Optimiza Debuggi Monitoria 	OK Cancel ture 32 bit C Compiler with Debug Opt sessor ies ation ing	Workspace File syste	em	Defaults Apply	

PPU project - add the SPU executable as an embed SPU input - step 3

- 1. Select **SPU** → **spu-gnu-debug** → **SPU**.
- 2. Click OK twice to return to the Properties for PPU window.



PPU project - configure additional settings

You can configure additional settings via the other tabs (for example, **Build steps**, **Binary parsers**, and so on), and categories (such as **Project References**).

Take a minute to explore these other tabs and categories, then click **OK**.

0		C/C++ · SPU/spu.c · Eclipse SDK			
<u>File</u> <u>E</u> dit Refa	3	Properties for PPU		0	2
t3+ G ta B	type filter text	Settings			++ "
Project Explo	Resource Builders ▼ C/C++ Build	Configuration: ppu-gnu32-debug	0	Manage configurations	fa □ □
Þ @ PPU Þ @ SPU	Discovery options	Tool settings PBuild steps Build artifact Binary par	sers o Error parsers		
	Tool chain editor	 ♥ BPU GNU 32 bit Assembler @ General ♥ BPU GNU 32 bit Embed SPU 	Embed SPU Inputs *\${workspace_loc:/SPU/spu-gnt	🕤 ඬ 😭 🔰 🖗 u-debug/SPU)*	g long)
	XL C/C++ Compil C/C++ General Project References Refactoring History	Architecture Architecture PU GNU 32 bit C Linker General Libraries)
	Run/Debug Settings	 Miscellaneous Shared Library Settings Profile & Trace Architecture 			
		 PPU GNU 32 bit C Compiler with Debug Options Preprocessor Symbols Directories Optimization Debugging 	2		(199)
	(š. m. 3)		Restore	Qefaults Apply	1
nº	Ø		ок	Cancel	

PPU project - create another new source file

- 1. Right click the **PPU** project.
- 2. Select New → Source File.

0	C/C	++ - SPU/spu.c - Eclipse SDK	le l
File Edit Refact	or Navigate Search Project Bun Window He	lp	
[m- 🖬 🖮]	g · B · C · G ·	第一時 日 - 第一 10 (10 - 10 - 10 - 10 - 10 - 10 - 10 -	■ C/C++ **
Project Explor	er 🛙 🔍 🖻 🕼 spu.c 🕱	- 0	🐉 Outline 🛱 💿 Make Ta 🔍 🗖
	😑 🤹 👻 #include <stdio.h></stdio.h>		1ª R X
D E PPU	New New	Project	🛯 stdio.h
Þ ଔ SPU	Open in <u>N</u> ew Window	🕃 File	🛯 profile.h
	Ctrl+C	😂 Folder	 main(unsigned long long
	■ <u>P</u> aste Ctri+V	🖹 Header File 👘	
	× Delete Delete	a Source File	
	Moye	63 Source Folder	
	Rena <u>m</u> e F2	🗟 C Project	
	import	E C++ Project	
	i⊴ Export	rt Example	c)
	Clean Project	🗂 Qther s 🐁 Remote Environments	0 % 🛃 🗇 😚 🔍 🗖
	Retresh F5		
	Close Unrelated Projects		
	Exclude from build		
	Build Configurations	×	
	Make targets		
	Index	>	
	Convert To		
	<u>R</u> un As	>	
	Debug As		
	Profile As	>	
	Team		
	Comp <u>a</u> re With		
	Restore from Local History		
😁 🐸 PPU	PDE IODIS		
	Properties Alt+Enter		

PPU project - enter the name for the new source file

In the **Source File** field, enter ppu.c, then click **Finish**.

3	C/C++ - SPU/spu.c - Eclipse SDK	_ 0 X
<u>File Edit Refactor Navigate Sea</u>	rch <u>P</u> roject <u>B</u> un <u>W</u> indow <u>H</u> elp	
13· 1 = 1 1 1 · 1 · 1 · 1 · 1 · 1	5- 8- 8- 0- 9- 0 10 10 8- 0- 9 0- 0-	E C/C++ *
Project Explorer 🛛 🔍 🗖	🗟 spu.c 🕱	😁 🗆 🐉 Outline 🕴 💿 Make Ta 😁 🗆
8 8 7	<pre>#include <stdio.h></stdio.h></pre>	14 N X 0 V
Þ 🖻 PPU	🔹 New Source File 🛛 🕅	u stdio.h
▶ ≝ SPU	Create a new source file.	 profile.h main(unsigned long long)
	Source Folder: PPU Browse	
	Source File: ppu.c	
	C	0 BH 25 B- 13+ 7 B
	Canad	
	<u>Linisn</u> Cancel	
		-
PPU 🗠		

PPU project - edit the source code file

```
Copy and paste the following source code into your editor, then save it (Ctrl+S):
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <libspe2.h>
#include <pthread.h>
extern spe_program_handle_t SPU;
#define SPU THREADS
                        8
void *ppu pthread function(void *arg) {
  spe_context_ptr_t ctx;
  unsigned int entry = SPE_DEFAULT_ENTRY;
  ctx = *((spe context ptr t *)arg);
  if (spe context run(ctx, &entry, 0, NULL, NULL, NULL) < 0)
{
    perror ("Failed running context");
    exit (1);
  }
  pthread_exit(NULL);
}
int main()
{
  int i;
  spe_context_ptr_t ctxs[SPU THREADS];
  pthread_t threads[SPU_THREADS];
  /* Create several SPE-threads to execute 'SPU'.*/
  for(i=0; i<SPU_THREADS; i++)</pre>
    /* Create context */
    if ((ctxs[i] = spe_context_create (0, NULL)) == NULL)
      perror ("Failed creating context");
      exit (1);
    /* Load program into context */
    if (spe_program_load (ctxs[i], &SPU))
      {
      perror ("Failed loading program");
      exit (1);
      }
    /* Create thread for each SPE context */
    if (pthread create (&threads[i], NULL,
    &ppu_pthread_function, &ctxs[i]))
      perror ("Failed creating thread");
      exit (1);
      }
  }
  /* Wait for SPU-thread to complete execution. */
  for (i=0; i<SPU_THREADS; i++)</pre>
    {
      if (pthread_join (threads[i], NULL)) {
      perror("Failed pthread_join");
      exit (1);
    }
    /* Destroy context */
```

```
if (spe_context_destroy (ctxs[i]) != 0) {
    perror("Failed destroying context");
    exit (1);
    }
    printf("\nThe program has successfully executed.\n");
    return (0);
}
```



Cell/B.E. simulator - create the local Cell/B.E. simulator

Objectives: After you have completed this part of the tutorial you will know how to:

- Create an environment
- Configure and start the simulator

Now that the projects are created and properly configured, you can test your program, but before you do this, you must create and start a Cell/B.E. environment. IDE integrates IBM's full-system simulator for the Cell/B.E. processor into Eclipse, so that you are only a few clicks away from testing your application on a Cell/B.E. environment, even if you do not have access to a physical Cell/B.E. processor.

In the **Remote Environments** view at the bottom:

- 1. Right click Local Cell Simulator
- 2. Select Create.



Cell/B.E. simulator - configure the simulator

This is the **Local Cell Simulator** properties window, which you can use to configure the simulator to meet any specific needs that you might have. You can modify an existing Cell/B.E. environment's configuration at any time (as long as its not running) by right clicking the environment and selecting **Edit**.

- 1. Enter a name for this simulator (for example, Local Cell Simulator).
- 2. Click the **Simulator** tab.

a	C/C++ - PPU/ppu.c - Eclipse SDK		_ 0 X
File Edit Refactor Navigate Searc	Target Environment Configuration		
C · C · C · C · C · C · C · C · C · C ·	Local Cell Simulator Properties for launch Local Cell Simulator		
	Target name: Local Cell Simulator Hardware Simulator Connectivity Launch Hardware Configuration Processor Type: Single BE © Aprox. RAM (MB) [256 Root file System Storage is: Non-persistent Journal File Kernel: [/opt/ibm/systemsim-cell/images/cell/vmlinux Root file system: [/opt/ibm/systemsim-cell/images/cell/sysroot_disk More	© Browse Browse Browse e options	A stdlib.h s
		uncer	
	Target Environment		Status
	Attached Cell Simulator		
	Generic Host		
	Local Cell Simulator		
	Remote Cell Simulator		
D.			
Cell/B.E. simulator - configure the simulator options

In the **Simulator** tab, check the box labeled **Show TCL console**, then click **Finish**.

0	C/C++ • PPU/ppu.c • Eclipse SDK	
File Edit Refactor Navigate Search	Target Environment Configuration	
11- 12 a 12 d- 61- 2- 6- 4	Local Cell Simulator	□ □ C/C++ *
🗞 Project Explorer 🛚 🔍 🔍 🗖	Properties for launch Local Cell Simulator	🗝 🗆 🕃 Outline 🛛 💿 Make Ta 🔍 🗆
<pre></pre>	Target name: Local Cell Simulator Hardware Simulator Connectivity Launch Simulator Options Show simulator control window Ø Show Show Show Cluster control window Show Linux console Ø Show TCL console Show TCL console Ø Use default working directory Browse Working directory: [Opt/ibm/systemsim-cell/run/cell/linux] More options More options	C If R R r r r r r r r r r r r r r r r r r
	© Einish Cancel □ Problems a lasks a console a Properties a remote Environments a	
	Target Environment	Status
	Attached Cell Simulator	
	Generic Host	
	Local Cell Simulator	
	Remote Cell Simulator	

Cell/B.E. simulator - start the simulator - step 1

- 1. Click the + next to Local Cell Simulator.
- 2. Select the new simulator environment.
- 3. Click the **Start the Environment** (green arrow) button.

3	C/C++ - PPU/ppu.c - Eclipse SDK	
<u>File</u> Edit Refactor Navigate	Search Project Bun Window Help	
] C1+ (a) (a) (a) + (3+] \$+ \$+ [\$+ \$+ \$+ \$+ \$= \$ \$= \$ \$= \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$	E
Project Explorer 2		Coutline ≈ Make Ta Coutline ≈ A Nake Ta
<pre>▼ ≤> PPU</pre>	<pre>></pre>	<pre>u stdlib.h u stdlib.h u erno.h p libspe2.h u pthread.h SPU_THREADS ppu_pthread_function(vo main(): int</pre>
	E Problems @ Tasks @ Console Properties Remote Environments #	Start the Environment
	Accached Cen Simulator	
	Generic Host	
	E Local Cell Simulator	
	Local Cell Simulator	Stopped
	Kemote Cell Simulator	
0°		

Cell/B.E. simulator - start the simulator - step 2

The simulator now begins to launch (this can take a few minutes).

a	C/C++ - PPU/ppu.c - Eclipse SDK	
<u>File Edit Refactor Navigate Sear</u>	ch <u>P</u> roject <u>B</u> un <u>W</u> indow <u>H</u> elp	
1 Cl + 🗟 💩 📓 🔞 + 🚳 + 🗟 + 🞯 + 🗍 4	in ⊗r φ· Q· Q· @ # ∃, ⊕ β· δ· φ α· α·	₽ QC/C++ *
Project Explorer x	© spuc № ppuc №	Outline Make Ta Outline Set
► Contractions ► Contractions ► Solutions	/* Wait for SPU-thread to complete execution. */ for (i=0; iSPU THREADS; i++) { Start the Environment Complete Environment Complete Environment Creating the machine	<pre>u stollo.n u stollo.h u errno.h // libspe2.h u pthread.h SPU: spe_program_hand SPU_THREADS o ppu_pthread_function(vo main(): int</pre>
	Run in Background Cancel Details >>	
	Problems Tasks Console R Properties Remote Environments Linux console for Local Cell Simulator	
	Start th	e Environment: (7%) 🛛 🖘 🦉

Create an application launch configuration

Objectives: After you have completed this part of the tutorial you will know how to:

• Create and configure a C/C++ Cell Application launch configuration

Next, you need to create and configure a C/C++ Cell Application launch configuration.

To do this, click **Run → Open Debug Dialog**.

3	C/C++ - PPU/ppu.c - Eclipse SDK	
<u>File Edit Refactor Navigate Se</u>	earch Project Bun Window Help	
10-00 0 0 0 0-00-0-0-	Stevensor Shift+Ctrl+B	≣ RC/C++ *
🕒 Project Explorer 🛛 👘 🕯	G Spu.c Toggle Method Breakpoint Toggle Watchpoint Toggle Watchpoint	Dutline 🛛 🖲 Make Ta 🐃 🗖
♥ 실 PPU > & Binaries > ≥ Includes > ≥ ppu-gnu32-debug > ≥ spu_objs > ≥ ppu.c > ≥ SPU	<pre>} Give a set of the set of t</pre>	<pre>u stdlib.h u stdlib.h u errno.h M libspe2.h u pthread.h o SPU : spe_program_hand s SPU_THREADS o ppu_pthread_function(vo o main() : int ulator!)</pre>

Application launcher configuration - create a new C/C++ Cell/B.E. target application configuration

In the left pane:

- 1. Right click C/C++ Cell Target Application.
- 2. Select New.



Application launcher configuration - modify the debug configuration

In the **Main** tab of the debug configuration, make sure that project **PPU** is in the **Project** field.

1.3	9	Debug		D BOC/CTT
 Project Explor PPU Binaries 	Create, manage, and run o Program not specified	onfigurations	-	he x ● Make Ta □ □ i ⁴ ₂ x x ³ • ▽
▶ Ø Includes	0 8 8 8 3-	Name: PPU ppu-gnu32-debug]	dio.h
 ▷ ppu-gnus. ▷ spu_objs ▷ di ppu.c ▷ SPU ▷ SPU ▷ SPi Includes ▷ spu-gnu-d ▷ spu.c 	type filter text □ C/C++ Attach to Local ■ & C/C++ Attach to Local > & C/C++ Cell Target Appl □ C/C++ Local Applicatio □ C/C++ Postmortem de ■ Eclipse Application □ GDB Hardware Debugg ■ Java Applet □ Java Applet Java Application Julit JUnit Plug-in Test	Main Target Launch 🛱 Synchronize 🖷 Environment 🗞 Debugger 🖗 Sour Project: PPU C/C++ Application: Search Project C/C++ Connect process input .output to a terminal.	Browse	rno.h spe2.h hread.h U : spe_program_hand U_THREADS u_pthread_function(vo ain() : int
	© GRemotic Java Applicatio Filter matched 14 of 14 item (1	Apply Debug (Revert Close	ator!)

For the C/C++ Application field, click Search Project.

Application launcher configuration - select C/C++ application

The **Qualifier** section lists the different build configurations that can be used (32 or 64 bit gnu, xlc, and so on). The **Binaries** section lists the available binaries for the corresponding **Qualifier**. For the **PPU** project, you have not changed the default build configuration, so only one **Qualifier** is currently available.

Select the **PPU** binary and click **OK**.

0	1	Program Selection			[_][n][x
Eile Edit Refac	or Navigate Search Project Be	Choose a program to run:		í.	x) B RC/C++ *
Project Explor	Create, manage, and run com O Program not specified	Binaries:		Ť	ne ⊠ ⊕ Make Ta □ □ J ^a z > x² ● ▽ dlib b
 Includes ppu-gnu3: ppu-gnu3: ppu-gnu3: ppu-gnu3: spu_objs ppu-gnu3: spu_objs spu_	Image: Second		ebugger 🦆 So	urce າ	dio.h rno.h spe2.h hread.h ?U : spe_program_hand
 ▶ & Binaries ▶ Includes ▶ espu-gnu-d ▶ e spu.c 	C/C++ Local Applicatio C/C++ Postmortem de Eclipse Application GOB Hardware Debugg Java Applet Java Application Julnit Ellipit Plunip Test	Qualifier: • ppcbe - /PPU/ppu-gnu32-debug/PPU	rc <u>h</u> Project)	Browse	U_THREADS u_pthread_function(vo ain() : int
	OSGi Framework Remote Java Applicatio Filter matched 14 of 14 item	© OK Cancel	Apply	Revert	
<u>es</u>	(INFO:	The simulator is PAUSED.)	<u>реонд</u> у назні соншала.	- 110140 CINC 021	mulator!)

Application launcher configuration - select target environment

- 1. Navigate to the **Target** tab.
- 2. For the **Choose target** field, select the simulator that was just created, then go to the **Launch** tab.



Application launcher configuration - configure the Launch tab

Here you can specify command line arguments as well as bash commands that are executed before and/or after the Cell/B.E. application runs.

C/C++ - PPU/ppu.c - Eclipse SDK File Edit Refactor Navigate Search Project Bun Window Help 0 C/C++ " Make Ta 😑 🗆 Project Explorer 🛚 Create, manage, and run configurations [Debugger]: The CPU is not supported by selected debugger. R X⁵ 0 7 🗢 🖾 PPU 0 Binaries
 Includes Name: PPU ppu-gnu32-debug type filter text 🗿 Main Target Launch 🔍 😂 Synchronize 🧐 Debugger 💱 Source 🐴 ppu-gnu32-debug
 spu_objs C/C++ Attach to Local Application Arguments ☞ B C/C++ Cell Target Application Command line arguments: Þ 🖻 ppu ogram_hand A PPU DI SPU SPU C/C++ Local Application Þ 👯 Binaries Customize application launch: _function(vo C/C++ Postmortem debugger ▶ S Includes Output parser:: none \$ Eclipse Application spu-gnu-debug Bash commands, executed before application: C GDB Hardware Debugging ▶ @ spu.c 👳 Java Applet I Java Application Je JUnit Bash commands, executed after application Ji JUnit Plug-in Test OSGi Framework Remote Java Application SPU Timing (Std. Make Project) Apply Revert 0 - 13 - - D Filter matched 14 of 14 items 0 Close (INFU: The simulator is PAUSED. /PPU/ppu.c

Next, go to the **Synchronize** tab.

Application launcher configuration - specify resources to synchronize

In this tab, you can specify resources (such as input/output files) that need to be synchronized with the Cell/B.E. environment's file system before, or after the application runs or both.

Use **New upload rule** to specify resources to copy to the Cell/B.E. environment before the application runs, and use **New download rule** to copy resource(s) back to your local file system after the application has run.

Do not forget to check the **Upload rules enabled** and/or **Download rules enabled** boxes after adding any upload/download rules.

Now go to the **Debugger** tab.

<u>File Edit Refactor M</u>	lavigate Search Project <u>R</u> un <u>W</u> indow <u>H</u> el	p	
] C1 × 12 ≥ [16] 16 ×]	3	Debug	■ ■ C/C++ *
Project Explorer 🕱	Create, manage, and run configurations [Debugger]: The CPU is not supported by s	elected debugger.	● Make Ta □ □ J ^a z R X ^s + ▼
	□ ■ ■ = >-	Name: PPU ppu-gnu32-debug	
P @ Includes b @ ppu-apu32-deb	type filter text	🕒 Main Target Launch 😂 Synchronize 👒 Debugger 🖗 Source 🛰	h
 ppu-grussz-deb spu_objs B opu c 	C/C++ Attach to Local Application ✓ C/C++ Cell Target Application	Synchronize rules: Upload rules are executed before application starts. Download rules are	I.h be program hand
 ♥ discription ♥ discr	© C/C++ Cocal Application © C/C++ Cocal Application © C/C++ Postmortem debugger ● Eclipse Application © GDB Hardware Debugging Ø Java Applet © Java Application Java Application	executed arten application missies.	READS nread_function(vo int
	J∜ JUnit Plug-in Test	Upload rules enabled Download rules enabled	
	OSGi Framework Remote Java Application	New upload rule New download n Edit selected Remove selected	
	SPU Timing (Std. Make Project) Filter matched 14 of 14 items	Apply Revert	d 0- 0- °0
	0	Debug Close	6
□° 🗟 /PPU/ppu.	(INFO: INE SIMULATOR IS P	#USEU.)	

Application launcher configuration - select debugger and launch the debug configuration

In the **Debugger** field, you can choose from **Cell PPU gdbserver**, **Cell SPU gdbserver**, or **Cell BE gdbserver**.

To debug only PPU or SPU programs, select **Cell PPU gdbserver** or **Cell SPU gdbserver**, respectively. The **Cell BE gdbserver** option is the combined debugger, which allows for debugging of PPU and SPU source code in one debug session.

- 1. Set the **Debugger** field to **Cell BE gdbserver**.
- 2. Click **Debug** to launch the debug session.

•		C/C++ - PPU/ppu.c - Eclipse SDK			(_)(¤)(X
Eile Edit Refa CI ← G ← G ← Project Expl ♥ G PPU ▶ & Binaries	ctor Navigate Search Project Bun Window Create, manage, and run configurations	Help Debug Name: PPU ppu-gnu32-debug		¢	C/C++ *
 S Includes B ppu-gnu S spu_objic S spu_objic S SPU S SPU S SPU S SPU S includes S spu_gnu S spu_c 	type filter text C/C++ Attach to Local Application C/C++ Attach to Local Application C/C++ Cell Target Application C/C++ Postmortem debugger Eclipse Application GDB Hardware Debugging Java Applet Java Applet Java Applet Junit JUnit Hug-in Test OSGi Framework Remote Java Application SPU Timing (Std. Make Project)	Main Target Launch & Syn Debugger: Cell BE gdbserve Stop on startup at: main Debugger Options Main Shared Libraries Com GDB debugger: // GDB command file: (Warning: Some commands operation of the debugger, GDB command set: St Protocol: m Verbose console mode	In this file may interfere with the for example "run".) andard (Cell BE) Apply	Common	Jram_hand
* @ /Pf	© PU/ppu.c		Debug	Close	

Debug the application

Objectives: After you have completed this part of the tutorial you will know how to:

• Use the debugger functions to step through your source code

The debug perspective functions in a similar way to any other GUI debugger you have may have used in the past. You can use it to step through your Cell/B.E. application's source code, while you monitor variables and register values, so you can find and fix bugs faster.

a Debug - PPU/ppu	.c - Eclipse SDK		
<u>File Edit Refactor Navigate Search Project Bun Window Help</u>			
C+ B B B B+ O+ Q+ Ø ≠ 8+ 0+ 9 ↔ 0+		e 19	Debug "
🕸 Debug 🛛 👘 🤞 🖗 🖬 🔳 😥 🔊 🖷 🖬 🛫 😁	🗆 💀 Variables 🛿 💊 Breal	akpoints 🎟 Registers 🛋 Modules	- 0
 ♥ BPU ppu-gnu32-debug [C/C++ Cell Target App Terminate ♥ Cell BE gdbserver gdb/mi (4/24/08 5:47 PM) (Suspended) ♥ Thread [1 cell] (Suspended: Breakpoint hit.) ■ 1 main() /home/danielhb/workspace_testing/PPU/ppu.c:33 0x1000175 ■ Connection to target ■ Remote process 	Name ⇔i > etxs > ethrases	Value -134340160 Oxfre6f44c Ovfre6f46c	
/opt/cell/toolchain/bin/ppu-gdb (4/24/08 5:47 PM)			3
homo/danialhh/worksnaco_torting/DDU/nou_anu22_dobug/DDU/A/2A/08_5	(*) [C		
<pre>int mein() { int i; spe_context_ptr_t ctxs[SPU_THREADS]; pthread_t threads[SPU_THREADS]; /* Create several SPE-threads to execute 'SPU'. */ for(i=0; i<spu_threads; ("failed="" ((ctxs[i]="spe_context_create" (0,="" *="" context="" context");<="" create="" creating="" i++)="" if="" null))="NULL)" perror="" td="" {=""><td></td><td> SPU_THREADS errno.h libspe2.h pthread.h stdio.h stdio.h SPU : spe_program_hand main() : int </td><td>le_t id*) · voi ♥</td></spu_threads;></pre>		 SPU_THREADS errno.h libspe2.h pthread.h stdio.h stdio.h SPU : spe_program_hand main() : int 	le_t id*) · voi ♥
Console 🛚 🤗 Tasks 🖭 Problems 🟮 Memory		- × % & 5 5 5 5 5	- 13 0
PPU ppu-gnu32-debug [C/C++ Cell Target Application] /home/danielhb/workspace	e_testing/PPU/ppu-gnu32-de	lebug/PPU (4/24/08 5:47 PM)	

Debug the application - switch to the C/C++ perspective

Switch to the C/C++ perspective. To do this:

- 1. Click the double right arrows at the top right
- 2. Select C/C++.

3	Debug - PPU/ppu.c	- Eclipse SDK	(J)));
Eile Edit Refactor Nav	rigate Search Project Bun Window Help		
th• B ≥ B \$• 0•	Q. 0 4 91-01- 0 0-0-		🖹 🌣 Debug 🛸
to Debug 🛙	NOPER XOLE NOT	💀 Variables 🕴 🎭 Breakpoints	E vis E ECC/C++
	ppu-gnu32-debug [C/C++ Cell Target Application]	Name	Value 😵 Java Srowsing
al <terminated. exit<="" td=""><td>t value: 0>Connection to target</td><td></td><td>Other</td></terminated.>	t value: 0>Connection to target		Other
<pre>cerminated, exi </pre>	t value: 0>Remote process		
<pre>cterminated, exit</pre>	t value: 0>/opt/cell/toolchain/bin/ppu-gdb (4/24/08 5:47 PN		
🚽 <terminated, exi<="" td=""><td>t value: 0>/home/danielhb/workspace_testing/PPU/ppu-gn</td><td></td><td></td></terminated,>	t value: 0>/home/danielhb/workspace_testing/PPU/ppu-gn		
a	())))	0	
🗟 ppu.c 🛿 🗟 spu.c			🗝 🛛 🔮 Outline 😫 🛛 🞼 🍇 🦿 🔹 🗖
<pre>int main() { int i; spe_context_ptr_t pthread_t threads /* Create several */</pre>	<pre>ctxs[SPU_THREADS]; [SPU_THREADS]; .SPE-threads to execute 'SPU'. READS; i++) { xt */ spe_context_create (0, NULL)) == NULL) { ed creating context');</pre>		<pre># SPU_THREADS # errno.h # libspe2.h # pthread.h # stdio.h # stdio.h # stdio.h \$PU: spe_program_handle_t # main() : int # onpu othread function(void*) : voi </pre>
🗉 Console 🛱 🧔 Tasks 🛙	1 Problems		= x % k s s s s
<terminated> PPU ppu- Process /tmp/IBM/Cell Listening on port 100 Remote debugging from Killing inferior</terminated>	<pre>gnu32-debug [C/C++ Cell Target Application] Remote proce IDE/arthas.ltc.br.ibm.com_danielhb/PPU/12090699674 00 i host 172.20.20.1</pre>	⊧ss 65/PPU created; pid = 663	
C.			
A CONTRACT OF			

Dynamic profiling tool

Objective: After you have completed this part of the tutorial you will know how to:

- Include the profiling functions in the spu.c code
- Run and view the results of dynamic performance analysis

Next, you see how to use the dynamic profiling tool on the SPU code.

- 1. Open the editor for **spu.c**.
- 2. Uncomment the 3 lines prof_clear();, prof_start();, and prof_stop();.
- 3. Rebuild the project by saving the source file (Ctrl+S).



Dynamic profiling tool - change SPU modes in the simulator window

In order to use the dynamic profiling tool, the SPE's must be in **pipeline** mode.

Currently the SPE's are running in **fast** mode, so you need to change this.

- 1. Open the Simulator GUI window (systemsim-cell),
- 2. Click the button labeled **SPU Modes**.

0		C/C++ - SPU/s	pu.c - Eclipse SDK				(_)(x)
<u>File Edit Refactor Navigate</u>	e Se <u>a</u> rch <u>P</u> roject <u>B</u> un <u>W</u> i	ndow <u>H</u> elp					
0- 11- 12 2 2 2 2 2 2 3	· @• @• %• @• \$• 0	• 9.• 9.• @ # =	1 ID 1) - 01 - 1 2 0	p+ Q+			≅ [©] C/C++ "
Project Explorer #	🤹 🔻 🗝 🖬 🗟 ppu.c 🛛 🗟 s	pu.c 🛚			• 0	BE Outline 18	💿 Make Ta 😁 🗆
 ✓ SPPU ▶ Sincludes 	#include <s #include <s int main(ur</s </s 	stdio.h> profile.h> usianed lona lona i	d) systemsim-cell		ĥ	별 profile. 별 stdio.h	i k x • ▼ h
P @ ppu-ghu52-debug P @ spu obis	File Window					Help in(u	nsigned long long)
 ▶ @ ppu.c ▼ SPU ▶ # Binaries 	B mysim B PPE0:0:0 B PPE0:0:1	Advance Cycle Ar	base nount : 1	• Cycles: 1,612,6	90,067		
▶ Includes	B C SPEO	Advance Cycle	Go	Stop	Service GDB		
Þ 😑 spu-gnu-debug	B SPE1	Triggers/Breakpoints	Update GUI	Debug Controls	Options		
▶ Espu.c	B C SPE3	Emitters	Mode	SPU Modes	SPE Visualizatio	on	
	B C SPE4	Process-Tree	Process-Tree-Stats	Track All PCs	Event Log		
	B D SPE6				Exit		
	SHE' Load-Elf-App Load-Elf-Kernel MemoryMap SystemMemory Stats Linux console f [root@(none)	or Local Cell Simulato	r ,		2) 1
	[root@(none) [root@(none) (INFO: The	<pre>tmp]# Configuratio tmp]#(INFO: Use t simulator is PAUSE</pre>	n complete his console to w D.)	rite Linux/Bash c	ommands inside t	the simulato	r!) •

Dynamic profiling tool - change the SPEs to pipeline mode

To change all of the SPE's to **pipeline** mode, click the **Pipe** button at the bottom, then go back to Eclipse.



Dynamic profiling tool - launch the PPU application

Now that you have included the profiling functions in the **spu.c** code, and have the SPE's set to **Pipeline** mode, you can launch the application.

Ctrl+F11 F11	ov ⇔	E CC/C++ " C Outline X Make Ta C Make Ta C
Ctrl+F11 F11	¢× ⇔-	■ Dutline X Make Ta ■ D Make
Ctrl+F11 F11	소 1 PPU ppu-gnu32-debug	Boutline X Make Ta Dutline X Make T
Ctrl+F11 F11	(b 1 PPU ppu-gnu32-debug	u profile.h stdio.h main(unsigned long long
Shift+Ctrl+B	emote Environments	
tor ion complete this console to w SED.)	write Linux/Bash commands insi	de the simulator!)
	> Shift+Ctri+B t tor complete this console to v SED.)	Shift+Ctrl+B t tor ion complete this console to write Linux/Bash commands insi SED.)

To do this, click **Run → Run History → PPU**.

Dynamic profiling tool - view the results of the dynamic performance analysis

You can view the results of the dynamic performance analysis.

To do this:

- 1. Open the Simulator GUI window.
- 2. Expand the category SPE0
- 3. Open SPUStats.

ile Edit Befactor Navigate Se	arch Project Bun Window Help	a otabare relibre and		(2)(2)
Late Heractor Havigate Se	systemsim-cell	21 I	_ 0 x	
File Window				
B results PPE0001 PPE001 PPE001 PPE001 PPE001 PPE001 PPE001 PPE001 Adva PPE001 Adva PPE001 PPE001 Adva PPE001 PPE001 Adva PPE001 PP	SPU DD3.0 Total Instruction count Total Instruction count Performance Cycle count Performance Enstruction count Performance CP1 Branch instructions Branch Atken Branch instructions Stoperations (MADDs=2) DF operations (MADDs=2) Total cycle	38833 2161 17.97 3224 557 (548) 5.79 (5.88) 75 1 210 0 Store and Prefetch 66 ict target is on each instruction class dependency stalls dependency stalls bendency stalls	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	S Outline S Make Ta S Outline S Make Ta S Outline S O

SPU timing tool - static timing analysis

Objectives: After you have completed this part of the tutorial you will know how to:

- Launch the SPU Timing tool
- View data collected by the SPU Timing tool

To launch the **SPU Timing** tool:

- 1. Right click spu.c
- 2. Select SPU Timing > Launch SPU Timing with given parameters



SPU timing tool - view the SPU timing output

You see the output from the SPU timing tool in the **Console** view.

Refresh your SPU project (right click SPU project, then click **Refresh**), to display a new directory named sputiming, which also contains the SPU timing output.

🖕 Project Explorer 🛛 📄 🍓 🔻 🗖 🗖	🗟 ppu.c 🕱 🗟 spu.c 🤍 🖱 c	🕽 🚼 Outline 🛛 💿 Make Ta
 ✓ SPU ▶ Sinaries ▶ Includes ▶ pu-gnu32-debug ▶ spu-objs ▶ gpu.c 	<pre>#include <stdib.h> #include <stdib.h> #include <stdib.h> #include <stdib.h> #include <strb.h> #include <pthread.h> #include <pthread.h< p=""></pthread.h<></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></pthread.h></strb.h></stdib.h></stdib.h></stdib.h></stdib.h></pre>	# SPU_THREADS # errno.h # libspe2.h
 ✓ ≦ SPU ▶ Inaries ▶ Includes ▶ espu-gnu-debug 	1411 stqd \$#r.16(\$sp) 1413 stqd \$sp.48(\$sp) 1414 ar \$sp.\$sp.48 1416 idd \$2,32(\$sp)	
₽ B spuk	1417 cob \$4.0(\$5) 1418 study \$2.3(\$cp) 1421 and \$0,\$0,\$0 1422 inop 1424 and \$30,\$0,\$0 1422 inop 1424 and \$30,\$0,\$30 1425 inop 1428 idd \$2.3(\$cp) 1428 idd \$2.3(\$cp) 1428 idd \$2.3(\$cp) 1428 idd \$2.3(\$cp) 1429 cot \$4.52.0 1430 brsi \$ir.printl 1432 and \$31,\$31 1435 il \$2.0 1435 il \$2.0 1438 at \$5,\$\$5,48 1439 idd \$ir.16(\$cp)	

Simulator console - toggle the simulator console view

Objective: After you have completed this part of the tutorial you will know how to:

• Switch between available console views

In the **Console** view toolbar, a small blue monitor icon is displayed. You can use this icon to toggle between the available consoles.

Click the down arrow to the right of the blue monitor icon.



Simulator console - select the simulator TCL console

Select the option TCL console for Local Cell Simulator.

3				
<u>File Edit Refactor Navigate</u>	Search Project Bun	Window Help		
11	• %+ 8+ \$+ 0 +	🐅 💁 🖉 🗿 🗃 🗇 🖓 🗉 🖉		≅ © C/C++ *
Project Explorer 🛙	. * * •	🗟 ppu.c 🛱 🗟 spu.c	- 0	🚼 Outline 🕱 💿 Make Ta 😁 🗖
 ✓ SPPU ▶ Sincludes ▶ Spu-gnu32-debug ▶ spu_objs 		<pre>#include <stdlib.h> #include <stdlib.h> #include <stdlib.h> #include <errno.h> #include tibspe2.h> #include <pthread.h> </pthread.h></errno.h></stdlib.h></stdlib.h></stdlib.h></pre>	×	# SPU_THREADS # errno.h # libspe2.h
▶ 🖻 ppu.c		🖺 Problems 🖨 Tasks 🖾 Console 😫 🖉 Pro	operties 🔥 Remote Environments 🔹 SP	U Timing Diagram 🗧 🗉
▼ 🗟 SPU		SPU Timing		R 2 2 0- 0-
 ▷ Includes ▷ Source debug 			○1 Linux console for ○2 TCL console for	or Local Cell Simulator
		8	. serti . Abyte . Abyte . Abyte . Abyte . 2byte . 2byte . Abyte . Abyte . Abyte . serti . Abyte . serti . Abyte . serti stri 	. debug_aranges, bxlc bx2 i.Lebug_info0 bx0 bx2 bx2 bx2 bx2 bx4 bx0 con .letxt0,.lext0 bx0 bx0 bx0 con .debug_macinfo for ".e g "long unsigned int" t "CCC: (GNU) 4.1.1"
D*				
1.5				

Simulator console - view the simulator's TCL console

This console can be used to pass TCL commands to the simulator.

The Fair Helactor Havidare Segren Fioleer Pa	. Шилом Пер	() () () () () () () () () ()
[C]+ [] (] [] [] (]+ 63+ []+ (] ≤ − [] ≤ − [] ≤ − [] ≤ − []	• Se• Se• [@ #] 3: 3: 3: 3: 5: 5: 5: 5: 5:	🖾 C/C++ *
🖕 Project Explorer 🛛 🛛 😑 🍇 🤻 🖶 C	🗟 ppu.c 🗷 🗟 spu.c 🧧 🗖	🕃 Outline 🕱 🛞 Make Ta 🖱 🕫
Project Explorer n	<pre>P ppuc # @ spuc finclude <stdio.h> finclude <ftdio.h> finclude <ftdio.h< finclude <ftdio.h> finclude <f< td=""><td>Coutline X Make Ta Coutline X Make Ta Coutline X Make Ta Coutline X Make Ta Coutline X Make Ta Superstand S Coutline X Make Ta Superstand S Coutline X Coutline X</td></f<></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h< </ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></ftdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></stdio.h></pre>	Coutline X Make Ta Superstand S Coutline X Make Ta Superstand S Coutline X Coutline X

Simulator console - use the simulator's Linux console

You can execute Linux commands inside the simulator (as long as the simulator is not stopped or paused).

In the **Console** view, switch to the console: **Linux Console for Local Cell Simulator**.

<pre>File Edit Refactor Navigate Search Project Bun Window Help To Fine Explorer B Project Explorer B Proje</pre>	3		C/C++ - PPU/ppu.c - Eclipse SDK			
Image: Set Get Get Get Get Get Get Get Get Get G	<u>File Edit Refactor Navigate</u>	Search Project Bun	<u>W</u> indow <u>H</u> elp			
<pre>Project Explorer # ● @ ♥ ● □ @ ppuc # @ spuc # @ spuc # # spuc # # SPU_THREADS # Binaries # ppu-gnu32-debug # spu_chis # ppuc # # SPU_THREADS # spu_chis # spu_chis # spu_chis # spu_chis # spuc # Problems @ Tasks © Console # Properties & Remote Environments @ SPU Timing Diagram ● C # SPU # Binaries * Include < this ppuc for Local Cell Simulator # spu_chis * spu_chis</pre>	C3+ 10 10 10 10 + 63+ 12 + 6	g• ≤ - ⊗ - \$• O•	δγ• 💁 🖉 11 10 2+ 3+ το φ+ φ+			₿ ₽C/C++ ·
<pre>SPU #include <stdii.h.> #include <stdii.h.> #include <stdii.h.> #include <stdii.h.> #include <stron.h.> #include <stron.h< th=""><th>Project Explorer</th><th>8 🕸 🗸 🖷 🖬</th><th>🗟 ppu.c 😫 🖉 spu.c</th><th>- D</th><th>E Outline 🕫</th><th>🖲 Make Ta 🔤 🗖</th></stron.h<></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stron.h.></stdii.h.></stdii.h.></stdii.h.></stdii.h.></pre>	Project Explorer	8 🕸 🗸 🖷 🖬	🗟 ppu.c 😫 🖉 spu.c	- D	E Outline 🕫	🖲 Make Ta 🔤 🗖
<pre>> @ Includes > @ spu-gnu-debug > @ spu-gnu-</pre>	 ✓ SPU ▶ Sinaries ▶ Includes ▶ Includes ▶ Includes, ▶ Includes, ▶ Includes, ▶ Spu_objs ▶ Includes, ▼ SPU ▶ Stimaries 		<pre>#include <stdlib.h> #include <stdlib.h< tdlib.h=""> #include <stdlib.h> #include <stdlib.h< td="" tdl<="" tdlib.h<=""><td>ote Environments</td><td># SPU_THF # errno.h # libspe2.h U Timing Diago</td><td>READS</td></stdlib.h<></stdlib.h></stdlib.h<></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></stdlib.h></pre>	ote Environments	# SPU_THF # errno.h # libspe2.h U Timing Diago	READS
	 ▶ Includes. <li< td=""><td></td><td><pre>a/, plugins/com.jbm.celldt.environment.cellsimulat (root@(nome) - J# modprobe systemsim met ATAL: Module systemsim net not found. (root@(nome) - J# iconfig eth0 172.20.20.20.retmas (root@(nome) - J# export CELLDT_NET_IP_MOST=172.20 (root@(nome) - J# export CELLDT_NET_IP_MOST=172.20 (root@(nome) - J# export CELLDT_NET_IP_MOST=172.20 (root@(nome) - J# export CELLDT_NET_IP_MOST=272.20 (root@(nome) + mpJ# clmod ux configure.sh (root@(nome) tmpJ# clmod ux configure.sh (root@(nome) tmpJ# clmod user configure.sh (root@(nome) tmpJ# clmod user user: assawd: Success Started sshd server (root@(nome) tmpJ# sty -echo (root@(nome) tmpJ# (Configure.sh (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (IVF): Use this console to wri (INF0: The simulator is PAUSED.)</pre></td><td>or/1209069797254 k 255.255.255.252 up 72.20.20.2 .20.1 S5.252 R/configure.sh > conf te Linux/Bash command</td><td>igure.sh s inside the</td><td>simulator!)</td></li<>		<pre>a/, plugins/com.jbm.celldt.environment.cellsimulat (root@(nome) - J# modprobe systemsim met ATAL: Module systemsim net not found. (root@(nome) - J# iconfig eth0 172.20.20.20.retmas (root@(nome) - J# export CELLDT_NET_IP_MOST=172.20 (root@(nome) - J# export CELLDT_NET_IP_MOST=172.20 (root@(nome) - J# export CELLDT_NET_IP_MOST=172.20 (root@(nome) - J# export CELLDT_NET_IP_MOST=272.20 (root@(nome) + mpJ# clmod ux configure.sh (root@(nome) tmpJ# clmod ux configure.sh (root@(nome) tmpJ# clmod user configure.sh (root@(nome) tmpJ# clmod user user: assawd: Success Started sshd server (root@(nome) tmpJ# sty -echo (root@(nome) tmpJ# (Configure.sh (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (Configure.sh) (root@(nome) tmpJ# (IVF): Use this console to wri (INF0: The simulator is PAUSED.)</pre>	or/1209069797254 k 255.255.255.252 up 72.20.20.2 .20.1 S5.252 R/configure.sh > conf te Linux/Bash command	igure.sh s inside the	simulator!)
	1°					

ALF API wizard

Objectives: After you have completed this part of the tutorial you will know how to use the ALF API wizard to:

- Create an ALF project
- Import the generated source code and configure the projects' properties to include the necessary libraries and to enable embedded SPU functionality
- Modify the ALF API wizard parameters
- Create and configure a new Cell/B.E. application launcher, so that the shared library project's .so file (libmy_alf_project.so) can be loaded during the remote application execution.
- Launch an ALF application

As of version 3.0, IDE includes a wizard which allows you to easily create a Cell/B.E. application that utilizes the Accelerated Library Framework (ALF) API. The ALF API provides a set of functions to a set of parallel problems on multi-core, host-accelerator type systems, such as the Cell/B.E. processor. Features of ALF include data transfer management, parallel task management, double buffering, and data partitioning. The next section of the tutorial walks you through an example use case of the ALF API Wizard.

To start using the wizard, click **File → New → Project**.

3					
File Edit Refactor Navigate	Search Project Bu	in <u>W</u> indow <u>H</u> elp			
New		🔊 C Project		P CICLL »	
Open File,		🖾 C++ Project		D HOCICTT	
Close	Ctrl+W	📑 Project		🛿 👫 Outline 🛛 💿 Make Ta 📄 🗖	
Close All	Shift+Ctrl+W	Convert to a C/C++ Make Project	P	1 × × • *	
Save	Ctrl+5	🚳 Source Folder		# SPU_THREADS	
🖬 Save As		😂 Folder		🖬 errno.h	
Save All	Shift+Ctrl+S	🗟 Source File		Iibspe2.h	
Revert		🖹 Header File		u pthread.h	
Move		📑 File	-	🖬 stdio.h	
Rename	F2	@ Class		u stdlib.h	
Refresh	F5	📬 Example		SPU : spe_program_hand	
Convert Line Delimiters To		ri <u>O</u> ther		main() : int	
الله <u>P</u> rint	Ctrl+P	t entry = SPE_DEFAULT_ENTRY;		oppu_pthread_function(vc	
Switch Workspace	1	a context of r t *largl;			
🖮 Import		text_run(ctx, &entry, 0, NULL, N	NULL, NULL) < 0) {		
🖾 Export		Failed running context");			
Properties	Alt+Enter	and the second se			
1 spu.c [SPU]		t(NULL);			
2 ppu.c [PPU]					
Exit		ks 🖸 Console 🗃 🗖 Properties 🐁 R	emote Environments SPU Timing Diagram		
	Linux console for	Local Cell Simulator			
	Started sshd se [root@(none) tm [root@(none) tm [root@(none) tm [root@(none) tm (INFO: The si	rver ip]# jp]# stty -echo jp]# Configuration complete jp]#(INFO: Use this console to w mulator is PAUSED.)	vrite Linux/Bash commands inside the sim	ulator!)	
	2	0000		3	
0°					

ALF API wizard - start the ALF IDE wizard

- 1. Expand the ALF for Cell category.
- 2. Select the Accelerated Library Framework (ALF) API Wizard option.
- 3. Click Next.

File Edit Refactor Navigate Search Project Bun Window Help Image: Search Project Explorer Bun Help Image: Search Project	 Amilies Robins 	C/C++ - PPU/ppu.c - Eclipse SDK	= n x
Image: Second	<u>File</u> <u>Edit</u> Refactor <u>N</u> avigate S	earch <u>P</u> roject <u>B</u> un <u>W</u> indow <u>H</u> elp	
Project Explorer x ■ puuc x @ snuc x @ snuc x @ Make Ta > @ PPU > @ shirtes > @ solve data parallel problems @ solve data parallel problems @ wireno.h @ libspe2.h > @ spu_gnu32.debug > @ spu_gnu32.debug Wizards: @ wireno.h @ libspe2.h > @ spu_c @ glava Project @ glava Project @ stdlib.h @ stdlib.h > @ spu_gnu.debug > @ General @ libract @ solution for Cell @ solution for Cell > @ spu_c @ ALF for Cell @ C < Back	t1+ 12 ≤ 18 @+ 63+ 61+ 64	4+ 8+ 8+ 0+ 9+ 4+ 8 4 1 1 10 1 1- 0+ 0 0+ 0-	≅ C/C++ *
▼ ● PPU ▶ © Binaries ▶ ⊚ Includes ▶ ⊕ ppu-gnu32.debug ▶ @ Includes ▶ @ ppu-gnu32.debug ▶ @ Includes ▶ @ Celerated Library Framework (ALF) API W/zard ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C ▶ @ C <td>🕒 Project Explorer 🛚 🔍 😁 🗖</td> <td></td> <td>🗢 🗆 🕃 Outline 🛛 💿 Make Ta 🔍 🗆</td>	🕒 Project Explorer 🛚 🔍 😁 🗖		🗢 🗆 🕃 Outline 🛛 💿 Make Ta 🔍 🗆
Back Next > Finish Cancel ram in al 2 0 - 0 - 0	<pre>© Project Explorer &</pre>	Neuroptic Select a wizard Create a new Cell BE application which utilizes the ALF API code template to solve data parallel problems Wizards:	# SPU_THREADS # erno.h # libspe2.h # pthread.h # stdio.h # stdio.h
LinStarted sshd server [rootRinone] tmp]#		Cancel Lin Started sshd server [root(none) tmp]#	
<pre>[root@inone) tmp]# stty -echo [root@inone) tmp]# Configuration complete [root@inone) tmp]#(INFO: Use this console to write Linux/Bash commands inside the simulator!) (INFO: The simulator is PAUSED.)</pre>		<pre>[root@inone] tmp # stty -echo [root@inone] tmp # Configuration complete [root@inone] tmp #(INFO: Use this console to write Linux/Bash commands inside the (INFO: The simulator is PAUSED.)</pre>	e simulator!)
0'	D*		

ALF API wizard - enter a project name

Enter a project name, then click Next.



ALF API wizard - change general parameters

Here you can change the ALF IDE wizard's general parameters. You can return to this page at any time if you need to make any changes.



Click Next.

ALF API wizard - add ALF buffers

This page displays any existing buffers, allows you to create new buffers, displays remaining SPU local memory, and displays the data transfer entry status. Click the corresponding **What's this?** text to learn more about the local memory and data transfer status sections.

Click the **Add** button to create a new buffer.

a state perfectore Manifester C	C/C++ + PPU/ppu.c + Eclipse SDK	×
	sarch Froiect Brin Million Heib Barch Froiect Brin Million Heib	≅ ℝ C/C++ *
Project Explorer 🛚 🔍 🖷 🗆	pnu c X R snu c New Project X	😁 🗆 🕃 Outline 🛛 💿 Make Ta 🔍 🗆
Image: Image	ALF Wizard Buffers Create the input/output buffers. Buffers: Name Type Add. Edit Remove Local Memory (accelerator)	<pre>SPU_THREADS # SPU_THREADS # errno.h # libspe2.h # pthread.h # stdlib.h # stdlib.h SPU : spe_program_hand * main() : int * ppu_pthread_function(vo</pre>
	India (204 KB): 251904 B Remaining: 251894 B (Wats: tips) Data Transfer (DT) Entries DT Status: N/A (Whats: tips) Ct	an a do-tro
	In Cancel	simulator!)
0*		

ALF API wizard - add the first buffer

Enter information in the fields as shown below, then click **OK**.



ALF API wizard - add a second buffer

Use the Add button to create another buffer, enter the parameters as shown below (use the same values as mat_a), then click OK.

3		C/C++ - PPU/ppu.c - Eclipse SDK		(_ s x
File Edit Refactor Navigate Search	h <u>P</u> roject <u>B</u> un <u>W</u> ir	Create New ALF Buffer		
] E3+ 🗟 🖨 🗍 📾 - 🚳+ 🗟+ 🞯+ 🗍 🛸	· · · · · · · · · ·	Manage		≅ RC/C++ **
Project Explorer 🛛 🔍 🖻 🖉 🖉 p	pu.c 22 la spu.c	mat_b	x	Image: Second
♥ SPPU ▶ & Binaries	ALF Wizard Buffe Create the input/e	Element type: float		# SPU_THREADS
 > Sincludes > Spu-gnu32-debug > ≥ spu_gnu32-debug > ≥ spu_gbis > ≥ ppu.c 	Buffers: Name mat_a	Element unit: ALF_DATA_FLOAT Buffer type: Input buffer	Type Input Buffer	Ibspe2.h phread.h stdio.h stdib.h
Binaries Sincludes e spu-gnu-debug	Add) Edit R	Dimension of array: 2-D \$ Size of dimensions: X: 512 Y: 1024 Z: 1		 sid: sp_program_nand main(): int ppu_pthread_function(vo
	Local Memory (ac Total (264 KB): 25 Remaining: 21 (What's_this?)	Distribution models: X: CYCLIC > Y: CYCLIC > Z: +		
1	Data Transfer (DT	Distribution sizes:		
E F Lin	DT Status: EQUAL (What's_this?)	X: 16 Y: 512 Z: 1		ram 🕞 🖬 🖆 🕒 🗇 👘
	٥	OK Cancel	Cancel	simulator!)
D*	L			

ALF API wizard - add the third buffer

Use the Add button to create the third and final buffer, enter the parameters as shown below (same values as mat_a and mat_b, except for the Buffer type), then click OK.



ALF API wizard - finish using the ALF IDE wizard

Click **Finish** to complete the wizard. The wizard automatically creates and configures the three new projects.

0	C/C++ - PPU/ppu.c - Eclipse SDK	(a)(x)
Eile Edit Refactor Navigate Searc	ch <u>P</u> roject <u>B</u> un <u>W</u> indow <u>H</u> elp	
C* 🗉 🖮 🗎 📓 🖓 * 🐼 * 🖉 * 🖉 *		≅ ©C/C++ *
Project Explorer X Project Explorer X Image: Comparison of the second	Description Description ALF Wizard Buffers Create the input/output buffers. Buffers: Name Name Type mat_a Input Buffer mat_b Input Buffer Add Edit Remove Output Buffer Local Memory (accelerator) Total (264 KB): 251904 B Remaining: 153500 B Windrs_ifiel Whats	Countine at the second se
ia: Lin Stri Irr Irr Irr	Data Transfer (DT) Entries DT Status: EQUAL (64) (What's Unit's U	ram R R R R R R R R R R R R R R R R R R R
D.		

ALF API wizard - ALF IDE wizard complete

The three new projects have now been created, the generated source code has been imported, and the projects' properties have been configured to include the necessary libraries and to enable embed SPU functionality. Looking at the source code, you notice that the ALF IDE wizard has generated code to handle many trivial tasks, such as input/output transfer list preparation, but you must write your own code for the computing kernel and data initialization.



ALF API wizard - Create new launch configuration

Finally, you must create and configure a new Cell/B.E. application launcher, so that the shared library project's .so file (libmy_alf_project.so) can be loaded during the remote application execution.



Click Run > Open Run Dialog...

ALF API wizard - add a new upload rule

- 1. Create a new **C/C++ Cell Target Application** configuration (like you did earlier in the tutorial).
- 2. Go to the Synchronize tab, then click New upload rule.

	C/C++ - ppu_my_alf_project/ppu_my_alf_project.c - Eclipse SDK	x
	avigate search Project Bun Window Heip	= = = = = = = = = = = = = = = = = = =
C Project Explorer	Create, manage, and run configurations Ø Program not specified	Make Ta = □ Make Ta = □ R × * * *
<pre>visitiony_alf_project visitiony_alf_project visitiony_alf_projet visitiony_alf_proje visitiony_alf_pr</pre>	Image: Symperity of the sympet in the sym	IN JF_SIZE h.h alf_project.h ge_name : char i_name : char i_name : char i_name : char i_name : char
Ispu_my_alf_pro	Upload rules enabled Download rules enabled	hame : char[]
	Filter matched 10 of 10 iten	Revert
	Image: Sum state of the state of t	Close
0*	Writable Smart Insert 161 : 37	
ALF API wizard - create a new upload rule - step 2

Make sure that the option **Use directory from launch configuration** is checked, then click the **Workspace** button.

0		C/C++ - ppu_my_alf_project/ppu_my_alf_project.c - Eclipse SDK			_ = x
Eile Edit Refactor N	avigate Se <u>a</u> r	ch <u>Project</u> Bun <u>Window</u> <u>H</u> elp			
C1+ 🗉 📾 🗎 🖬 🛛 🛍+ 6	a gu gu li	ξ× K)× th+ O+ O+ O ₊ × (th d) → (th d) → (th+ O+			× E C/C++ *
Project Explorer	Create, m	3	X		💿 Make Ta 🦳 🗆
 ✓ Slibmy_alf_project ▶ Sincludes ▶ ppu-gnu32-debi ▶ ppu-gnu32-debi ▶ ppu-gnu32-debi ▶ Sincludes 	 Program Type filter D C/C++ C C/C++ E Clipse Java A Jo JUnit JUJUnit OSGI F SPU Ti 	Upload Rule Please specify the remote directory and a list of files that shall be uploaded Remote directory: Use directory from launch configuration Remote directory Selected file(s): Add fi U W Remote Rem	load Rule asse specify the remote directory and a list of files that shall be uploaded. mote directory: Use directory from launch configuration mote directory. ected file(s): Add files: File(s) Directory Workspace Remove files:		iN JF_SIZE i.h _alf_project.h ge_name : char ip_name : char[i_name : char]
	© No	Options for all selected file(s): Readonly Download back if changed Preserve time attributes If file already exists: Overwrite OK thing to build for project ppu_my_alf_project	Cancel	Nove selected	name: char] float[][] float[][]
		Writable Smart Insert 161:37			

ALF API wizard - select the shared library .so file

- 1. Expand the Workspace and libmy_alf_project categories.
- 2. Click the **ppu-gnu32-debug** folder to view its contents.
- 3. Check the box for libmy_alf_project.so
- 4. Click OK twice to return to the Synchronize tab.

0			C/C++ - ppu_my_alf_project/ppu	_my_alf_project.c - Eclipse SDK			_ = x
<u>File</u> <u>Edit</u> Refactor <u>N</u>	avigate Se <u>a</u> r	rch Project	t <u>B</u> un <u>W</u> indow <u>H</u> elp				
C1+ ⊡ @ ⊕ @+ p	i a a l	6- 0- 0	- 0- 0- 0- 0 - 0 0 0 0	Run	_		× E C/C++ *
Project Explorer	Create, m	O he ent	cun configurations		×		🖲 Make Ta 🦳 🗆
	Program I	Upload R	ule				R X O T
✓ ibmy_alf_project	13 B X 1	Please specify the remote directory and a list of files that shall be uploaded.					IN JF_SIZE
👂 🔊 Includes	type filter	Remote c	Select the ressources from work	space to add			-
ppu-gnu32-debt	> b C/C++	⊠ Use di	🗢 🖃 🖾 Workspace	🗆 🕞 import_spu.mk		source "	n.h
> spu_objs	👌 ppu	Remote		■ liblibmy_alf_project.so		executed after	_alf_project.h
> ppu_my_an_proje	C/C++	Selected	🗢 🗏 🖨 ppu-gnu32-debug	🗆 🕞 makefile			
I common.h	 Eclipse Java Aj Java Jujunit Jujunit F 		spu_objs	G Sources mk			
ppu_my_alf_pro		020	□ ≝ ppu my alf project	D is sources.mk	H		
b & ppu_my_alf_pro			▷ □	:e		rge name - chai	
Spu_ny_an_proje					e		ip name : char[
👂 🍠 Includes	OSGi F				cted		:I_name : char[]
👂 💩 spu-gnu-debug	SPU II				L.C.U		hame : char[]
Ispu_my_alf_pro		Options I				move selected	hame : char[]
	Reado Downl		0	OK Cancel	i 👘		float[][]
	(m	If file alre				Revert	
	Filter match	in the dire	ady exists over mile				1 0 · 1 · · · ·
	0 0	Ø		OK Can	icel	Close)
	No	othing to	build for project ppu_my_alf_	project		1	-
0.	I		Writable	Smart Insert 161 : 37			

ALF API wizard - launch an ALF application

All that remains to be done is to check the box labeled **Upload rules enabled**. The libmy_alf_project.so file will now be copied to the working directory automatically before the application runs, so the PPU project executable can load it during runtime.

Click **Run** to begin running the application.

<u>ه</u>		C/C++ - ppu_my_alf_project/ppu_my_alf_project.c - Eclipse SDK	
<u>File</u> <u>Edit</u> Refactor	Navigate Search Project Bu	in Window Help	
13- 11 20 10 10	Br Br Gr Br Br Br Br	av Quy Quy (m)	📄 🗉 🗟 C/C++ 💾
Project Explorer	Create, manage, and run c	onfigurations	🛛 😢 Make Ta 🔍 🗆
 Glibmy_alf_proje gt Binaries gt Includes spu-Guigau32-di spu-guig2-di spu-guig2-di gt pu_my_alf_pr gt Includes gt Includes gt Includes gt pu_my_alf_guigau32-di gt common.h gt pu_my_alf_ gt pu_my_alf_ dt ppu_my_alf_ 			<u></u>
	X B 3+ Vye nitce text S ppu_my_aif_project p C/C++ Cell Target Appli C/C++ Local Application Eclipse Application Java Applet Java Applet Java Application Ju/Jinit Plug-in Test OSGi Framework	Name: [ppu_my_alf_project ppu-gnu32-debug Main Target Launch S Synchronize Environment Debugger Source Synchronize rules: Upload rules are executed before application starts. Download rules are executed after application finishes. UPLOAD 1 item From: libmy_alf_project/ppu-gnu32-debug/liblibmy_alf_project.so To: \${system_workspace}/\${user_workspace}/\${project_name}/\${timestamp}	LIGN BUF_SIZE hon.h h.h h h j.h j.h j.h d.h herge_name : chari etup_name : chari dtl_name : chari
Binaries Sincludes	Sto mining (Sta. Maker	☑ Upload rules enabled	I_name : char[]
> spu-gnu-debt		New upload rule New download rule Edit selected Remove selected	a : float[][]
·	Filter matched 10 of 10 item	Apply Reyert	
	٢	Bun Close	.pthread -
	Finished buildi	ing target: ppu_my_alf_project	_
0°		Writable Smart Insert 161:37	

IDE PDT plugin

Objective: After you have completed this part of the tutorial, you will know how to:

• Use the PDT to generate and view trace files

IDE provides an easy way to generate trace files using PDT - Performance Debugging Tools. First you must set PDT flags for your project. To do this, right click the project and choose **Properties**.

	Rei Hangate Begith Erojett Ha		Large Large and the second	
Designt Eurolan	<u>N</u> ew		1 m m lato no co co	B B B Outling
Project Explor	Open in <u>N</u> ew Window			
 Simple_spu_ Binaries Includes spu-gnu-d simple_sp 	Copy Paste Delete Renove from Context Moye Rename	Ctrl+C Ctrl+V Delete Shift+Ctrl+Alt+Down	ther as an array of ints or as a vector.	B stdie B spu_ ▷ @ (ano ● vec\ ● main
	≥ Import ≤ Export		ne variables to the 4 positions in vConst, int/(2, 2, 2, 2).	
	Clean Project Refresh Close Project Close Unrelated Projects	F5	wConst.myVec);	
	Exclude from build Build Configurations Make targets Index	>	, vz[i] = %d, vz[z] = %d, vz[s] = %d\n\n', Vals[2], v2.iVals[3]);	
- Comp	Convert To <u>R</u> un As <u>D</u> ebug As <u>P</u> rofile As <u>Team</u> Comp <u>are With</u> Restore from Local History PDE Tools	>) > > > >	perties B Remote Environments	

IDE PDT plugin - set up the C Linker flag

- 1. In the C/C++ Build, Settings category, go to C Linker options.
- 2. Select **Profile & Trace**.
- 3. Check Enable PDT compiler flags.

3		Properties for simple_s	pu_simd		
<u>File</u> <u>E</u> dit R	type filter text	Settings			
D~ C C	Resource Builders ▽ C/C++ Build	Configuration: spu-gnu-debug		Manage configurations	C++ " ce Ta □□
⊧ e ^s simple_	 C/C++ Build Discovery options Environment Settings Tool chain editor Variables XL C/C++ Compiler C/C++ General Project References Refactoring History Run/Debug Settings Task Repository 	 Tool settings Build steps Build artifact SPU GNU Assembler General SPU GNU C Linker General Libraries Miscellaneous Shared Library Settings Profile & Trace Overlay SPU GNU C Compiler with Debug Options Preprocessor Symbols Directories Optimization Debugging Warnings Miscellaneous Profile & Trace Architecture 	Binary parsers ○ Error parsers		
			Res	ore Defaults Apply)
0° ≌ s	Ø			OK Cancel	

IDE PDT plugin - set up the C compiler flag

To set up the C compiler flag, do the following:

- 1. Go to C Compiler with Debug Options.
- 2. Select Profile & Trace.
- 3. Check Enable PDT compiler flags.
- 4. Click **OK** to finish.



IDE PDT plugin - rebuild the project with PDT flags

Rebuild the project with PDT flags. The project must be built without any errors. Now you need to create a XML configuration file for PDT.

e	C/C++ - simple_spu_simd/simple_spu_simd.c - Eclipse SDK	_ + X
<u>File Edit Refactor Navigate S</u>	earch Project Bun PAPI Window Help	
0~ 0~ 0~ 0~ 0~ 0~	ଡିଦ୍ 💽 ୭୦୦ 🕸 ଦେ 🚱 ଦେ 🦑 🗇 🕼 🖉 🖉 ୬୦୦୦	
Project Explorer 🛚 🔍 📟 🗖	simple Build the active configurations of selected projects	🗖 🕃 Outline 🕱 💿 Make Ta 🦳 🗖
은 않 문 *	<pre>#include <stdio.h> #include <stdio.h> #include <stdio.h> #include <stdio.h> // Define a type we can look at either as an array of ints or as a vector. ty eat vals[4]; vector signed int myVec; vecVar; int main() { vecVar V1, V2, vConst; // define variables // load the literal value 2 into the 4 positions in vConst, vConst.myVec = (vector signed int){2, 2, 2, 2}; // load 4 values into the 4 element of vector v1 velvec = vector signed int {10, 20, 30, 40}; </stdio.h></stdio.h></stdio.h></stdio.h></pre>	H ₂ W V + P = P = P = P = P = P = P = P = P = P
	Problems @ Tasks © Console Properties Remote Environments	
	C-Build [simple_spu_simd]	
	<pre>make all Building file:/simple spu_simd.c Invoking: SPU GNU C Compiler with Debug Options spu_gcc OMFCIO TRACE -Dexit= pdt exit -DMain= pdt main -DLIBSYNC TRACE -I/opt/cell trace o0 -g3 -Wall -c -fnessage-length=0 -Winline -Wextra -fno-inline -mtune=cell Wno-main -marchccell -mess2 - MPO -M* -M* simple_spu_simd.d* -MT*simple_spu_simd.d* Sinshed building:/simple_spu_simd.c Building target: simple_spu_simd Invoking: SPU GNU C Linker spu-gcc -WI, -qW., -N* simple_spu_simd Finished building target: simple_spu_simd</pre>	/sysroot/usr/spu/include/ mfloat=fast -mdouble=fast - o*simple_spu_simd.o* "/ sr/spu/lib/trace/libtrace.a
😁 🧭 simple_spu_simd		

IDE PDT plugin - create a PDT configuration file

¢ C/C++	- simpl	le_spu_simd/simple_spu_simd.c - E	clipse SDK	_ + ×
<u>File</u> Edit Refactor <u>N</u> avigate Search <u>P</u> roject <u>B</u> un	PAPI)	Window Help		
New Shift+A	lt+N →	🖻 C Project	• dou :	■ C/C++ **
Open File _a		C++ Project	-	😫 Outline 🕴 🖲 Make Ta 📮
<u>C</u> lose 0	Ctrl+W	Project	回	R & S + 0
Close All Shift+0	Ctrl+W	Convert to a C/C++ Make Project		stdio h
M Save	Ctrl+S	Source Folder	ints or as a vector.	spu intrinsics.h
🗟 Save As		Source File		(anonymous)
Save All Shift+	Ctrl+S	Header File		vecVar : union
Revert		📑 File		main() : int
Moye		@ Class		
Rename	F2	📬 Example	a uConst	
Refresh	F5	Other	i vconst,	
Convert Line Delimiters to	,	- Other	•	
الله <u>P</u> rint	Ctrl+P	function		
Switch Workspace	>	<pre>(v1.myVec, vConst.myVec);</pre>		
im Import		nv2[0] = %d, v2[1] = %d, v2[2]	= %d, v2[3] = %d\n\n",	
i⊴ Export		als[1], v2.iVals[2], v2.iVals[3]	D;	
Properties Alt-	+Enter		251	
- Internet internet	- Lincol	ole 🛿 🔲 Properties 🐁 Remote Env	vironments	
1 test.xml [simple_spu_simd]				
2 simple_spu_sima.c [simple_spu_sima] 3 test c [test/src]				Í
4 array add.c [SoftCache PPU]		spu_simd.c		
Euit		iler with Debug Options	-DITESYNC TRACE - I/opt/cell/s	vsroot/usr/spu/include/
Lace -00 -go -man		fmessage-length=0 -Winline -Wext	ra -fno-inline -mtune=cell -m	float=fast -mdouble=fast -
Wno-main -march=ce simple spu simd.c	ell -mea	a32 -MMD -MP -MF"simple_spu_simo	i.d" -MT"simple_spu_simd.d" -o'	"simple_spu_simd.o" "/
Finished building	/si	mple_spu_simd.c		
😁 🐸 simple_spu_simd				

To create our XML configuration file for PDT, go to **File** \rightarrow **New** \rightarrow **Other**

IDE PDT plugin - open the wizard

To open the wizard, in the **Performance Debugging Tool** category, select **Configuration File for Cell**, then click **Next**.

0~ 0~ 0~ 0 0 0 0 0 0	V Gv %v ⊗v \$v Qv 9v Qv ⊅ G % 11 10 94 00 10 ⇔v ov	🗈 🖾 C/C++ 🛛 *
🔁 Project Explorer 🛚 🔍 🔍 🖤	Resimple cou cind.c s	🗖 📴 Outline 🛚 💿 Make Ta 🤗 🗖
E % ≥ *	Select a wizard This wizard creates a XML configuration file for the PDT tool using Cell B.E. as Wizards: Type filter text > © CVS > aya > Mylyn > © Performance Debugging Tool Configuration File For Cell > Plug-in Development © @ Back Next >	Image: Second
	🗄 Problems 🕢 Tasks 🗳 Console 🗷 🔅 Properties 🐁 Remote Environments	
	C-Build [simple_spu_simd] make all Building file:/simple_spu_simd.c Invoking: SPU GNU C Compiler with Debug Options spu-occ -DMPCTO TRACE - Dexit= pdt exit -DMain= pdt main -DLIBSYNC TRACE -I/opt/cell/	/sysroot/usr/spu/include/

IDE PDT plugin - enter a file name

Select a location and a name for the XML file, then click Next.

0	C/C++ - simple_spu_simd/simple_spu_simd.c - Eclipse SDK	_ + ×
<u>File Edit Refactor Navigate S</u>	Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>P</u> API <u>W</u> indow <u>H</u> elp	
0~] 1~ 🖬 👌 🗟 🛛 🖆~ 🖏 🖓	• @•] 《· 원·] \$•• Q• Q• Q•• Q•] @ @ # /] : @] 원· 원· 부· 수• 수·	E C/C++ ×
ြာ Project Explorer 🛚 🗧 🗖	simale enu cind e w	" 🗆 🔠 Outline 🛛 💿 Make Ta 🔍 🗆
Project Explorer ® 2 ■ 2	General Options New configuration file options Project and file name Project: isimple_spu_simd XML file name: pdt.xm[Courtine 31 @ Make la = = = = = = = = =
	<pre>weak all make all make. Nothing to be done for `all'.</pre>	
D.		

IDE PDT plugin - select event groups

In this window you can select which event groups your PDT session will capture. Select any event group and click **Next**.

3	C/C++ - simple_spu_s	imd/simple_spu_simd.	c - Eclipse SDK	
<u>File Edit Refactor Navigate</u>	Search Project Bun PAPI Window	<u>H</u> elp		
0~ 0~ 🖩 👜 🚔 📾 - 📴	~ @~ % ~ @~ \$~ Q ~ <u>9</u> ,~ Q ~ @	😂 🛷 🗊 🔯 🖗 🗸 🤪	¢ ⇔ ⇔	□ □ □ C/C++ *
Project Explorer x Project Explorer	Select available event group Select which event groups will b Available event groups:	pe included in the config	wuration file Selected event groups:	Contine ≈ Make Ta □ □ Intrinsics.h SuperVar: union
 ▶ ➡ spu-gnu-debug ▶ ➡ simple_spu_simd.c 	LIBALF_PERF LIBDACS MFCIO LIBDACS_PU2 LIBDACS_PERF LIBALF LIBDACS2 LIBALF_SPU LIBDACS_SPU	Add Remove Add All Remove All		• main() : int
	The second se	tk Next > md.c th Debug Options dt exit -DMain= pdt_ e-Tength=0 -WinTine D_MPMF*simple cou	Einish Cancel	<pre> a a a double fast - t -n*sime source of * / </pre>
] 0°	1	Writable Smart	Insert 12:26	

IDE PDT plugin - select sub-events

Next you need to fine tune your event selection by selecting events inside the events groups you chose before.

Click **Next** when you have finished.

Project Explorer		Dia Cutline 🛛 💿 Make Ta 🔍 🗆
 Simple_spu_simd Sincludes Sincludes Simple_spu_simd.c 	PDT Events Selection Select the events that you want to trace among. Events are organized in groups and subgroups. General Group Options © Enable PPE Profile © Enable SPE Profile Description © S LIBSPE_MAIL > E LIBSPE_MOGRAM_CONTROL > E LIBSPE_MFC_STATUS	Pre te te o o o o o o o o o o o o o o o o
	Endsh Cancel **** Build of configuration spu-gnu-debug for project simple_spu_simd **** make all make: Nothing to be done for `all'.	

IDE PDT plugin - set colors for events

In this last step, you select different colors for the events groups, which you chose on previous steps. This information is part of the PDT's XML configuration file and is used to generate PDT trace files. Click **Finish** to create the XML configuration file.

3	C/C++ - simple_spu_simd/simple_spu_simd.c - Eclipse SDK	2 + 3
<u>File</u> <u>Edit</u> Refactor <u>Navigate</u>	e Search Project Bun PAPI Window Help	
0~] * 🖩 🗁 🛱 🛣		➡ C/C++ **
Project Explorer 🛚 🗧		😁 🗆 🕃 Outline 🙁 💿 Make Ta 😁 🗆
v ésimple_spu_simd ≥ ∰ Binaries	Groups Position and Color Set position and color of each group	Image: static line Image: s
Includes	EventGroup name Vertical Ev Vertical Ev RGB color	Image: A constraint of the second
▷ ≥ spu-gnu-debug ▷ ② simple_spu_simd.c	2 LIBSPE2 0.0 0.0 0	 vecVar : union main() : int
	Select color C.E C.E C.E C. Cancel make all Building file:/simple spu_simd.c Invoking: SPU_GMU_C Compiler with Debug Options spu-gcc_DMFCIO_TRACE -Dexit=_pdt exit -DMain=_pdt_main -DLIBSYNC_TRACE -I/opt/cel trace -00 -g3 -Wall -c -fmessage-length=0 -Winline -Wextra -fno-inline -mture-cell Mno-main_marchreful -MmaX2_WMBHDHE_Simple sourcimel sourcimels sour	ll/sysroot/usr/spu/include/ - mfloat=fast - mdouble=fast - - orsimale sun sind of * /
D*	Writable Smart Insert 12:26	

IDE PDT plugin - modify the XML configuration file

- 1. Open the created XML file and set the **application_name** attribute to match the binary that will be used with PDT.
- **2**. Save the files.
- **3**. Press F5 to refresh the workbench.



IDE PDT plugin - using the Profile Launcher

You can now launch our application with PDT support. To do this:

- 1. Right-click the editor with the C file that will generate the executable.
- 2. Select **Open Profile Dialog**.

Ele Edit Refactor Navigate Search Pro	C/ Open Call Hierarchy oject Quick Outline	Ctrl+Alt+H		_ + ×
] O-] ⊡- III @ III @- Ø- I I I	© ✓ Source/Header	Ctrl+T	-T 🗈 🗟 C/C++ **	
🕒 Project Explorer 😫 👘 🗖 💽 simple	spu_si Show In	Shift+Alt+W >	- 0	🕃 Outline 🛛 💿 Make Ta 🔍 🗆
Image: simple_spu_simd // Def > Image: simple_spu_simd // Def	de <st cut<br="">de <sp copy<br="">ine a Paste</sp></st>	Ctrl+X Ctrl+C Ctrl+V	a vector.	intrinsics.h
▷ Sincludes ▷ Spu-gnu-debug ▷ c simple spu sind c ▷ vector s ▷ vector s ▷ vector s	if unic it iVal Source cctor \$ Refactor /ar;	>		 @ (anonymous) vecVar : union main() : int
pdt.xml int ma ve //	Declarations References Load Search Text	>		
vC // v1 // v2 //	I coad Show Concurrency Load Show Concurrency call Show IASTNode in DOM View .myVec Bun As intf (* Debug As	>] = %d\n\n".	
v2 re }	tivals <u>Profile As</u> clean Selected File(s) Build Selected <u>F</u> ile(s) Team	ہ د	☆ Open <u>P</u> rofile Dialog.	
L Problem C-Build [s	ms T Compare With Replace With	>		
Take all	Preferences		TRACE -I/opt/cell/sysroot/usr/spu/include/ Line -mtune-cell -mtloat=fast -mdouble=fast - molo courcied d = officient occurried of = //	
make att Building Invoking spu-gcc trace -0	file: Exclude from build : SPU -DMFCI Build Configurations 0 -g3 Make targets	>		
] 0*	A Remove from Context	Shift+Ctrl+Alt+Down		stante stat stall 11 / 2

IDE PDT plugin - launch the application with PDT support

- 1. Double left-click **C/C++ Cell PDT target application** to create a new launch configuration
- 2. In the **PDT** tab, select the option **Copy XML file to remote machine**. You need to configure:
 - Remote directory to copy the XML file must be a valid directory
 - · Local XML file. Select the configuration file you created on previous steps
 - Remote directory to create the trace files must be a valid directory
 - Prefix for trace file name any valid ASCII string
 - Local trace directory, where the generated trace files will be downloaded from the remote environment must be a valid directory

Click Profile to run the application with PDT support.

ile <u>E</u> dit Refac <u>t</u> o	÷	Profile	e	
O~ Cl~ 💷 👜 🖬 ⊳Project Explorer	Create, manage, and run configurat	I run configurations		
 Simple_spu_sit Simple_spu_sit Simple_spu_sit Distription <li< th=""><th>type filter text v C/C++ Cell PDT target application simple_spu_simd spu-gnu-debug Parallel Performance Analysis Performance Analysis</th><th colspan="2">Name: simple_spu_simd spu-gnu-debug Main Target Launch 🕿 Synchronize PDT PDT System Environment 🗆 Common Xml configuration file info ° Copy XML file to remote machine O Use XML file already on remote machine Remote directory where XML will be copied: //tmp/pdt Land/WL file</th></li<>	type filter text v C/C++ Cell PDT target application simple_spu_simd spu-gnu-debug Parallel Performance Analysis Performance Analysis	Name: simple_spu_simd spu-gnu-debug Main Target Launch 🕿 Synchronize PDT PDT System Environment 🗆 Common Xml configuration file info ° Copy XML file to remote machine O Use XML file already on remote machine Remote directory where XML will be copied: //tmp/pdt Land/WL file		
	Coal Anter Inte. //home/danielhb/runtime-New_configuration/simple_spu_simd/pdt.xml Browse Path to XML file on remote machine: //usr/share/pdt/config/pdt_cbe_configuration xml Trace file info Trace file info Remote directory where the trace file will be created: //tmp/pdt/trace Prefix for trace file name: Prefix for trace file name: Prefix for trace file name:			
		pdt_ Local trace directory: /home/danielhb/temp Browse		
	Filter matched 4 of 4 items			
	Ø	Profile Close	Runn	

IDE PDT plugin - check generated trace files

After the launch finishes, you can check the results of our profiling by opening the local directory set to receive the trace files generated.

These trace files can be viewed with the Eclipse-based VPA tool using the Trace Analyzer perspective, for example.

3	C/C++ - simple_spu_simd/simple_spu_simd.c - Eclipse SDK	
<u>File Edit Refactor Navigate</u>	Search Project Bun PAPI Window Help	
0~ 10~ 🖬 📾 🖬 10~ 8~ 3	▶ 중॰ š॰ 8॰ \$॰ 0॰ 9,॰ 9,॰ 1,॰ 1 8 6 4 11 10 9 ॰ 8 ॰ 5 ↔	■ C/C++ **
🕒 Project Explorer 😫 👘 🗖	🛾 🕼 simple_spu_simd.c 🛪 🕞 pdt.xml	🗝 🛛 🕃 Outline 🛛 💿 Make Ta 🔍 🗆
<pre>© Simple_spu_simd > @ Simple_spu_simd > @ Simple_spu_simd.c </pre>	<pre>sfinclude <spu_sincle <spu_intrinsics.h="" in="" picking="" sfinclude="" x=""> // Define a type we can look at either as an array of ints or as a vector. typedef union { int ivals[d triangle intervention] vector sign vet</spu_sincle></pre>	<pre></pre>
	<u>er</u>	

View the Eclipse preferences window

You can view and modify numerous cell environment settings in the **Preferences** window.



Click Window > Preferences...

View the IDE environment preferences

On the left side, expand the **Cell** category. Here you can change many different options regarding IDE including **Debug** options, **Cell Environment** settings, and **SPU Timing** binary location.

0	C/C++ - pp	u_my_alf_project/ppu_my_alf_project.c - Eclipse S	5DK	= e x
<u>File</u> Edit Refactor Navig	5			
113× 10 10 10 10 × 63× 1	type filter text	Cell	Ø* -	■ C/C++ *
Project Explorer # Project Explorer # Binaries Sicus pu_gnu32.debug bespu_gnu32.debug bespu_gnu_gnu32.debug bespu_gnu_gnu32.debug bespu_gnu32.debug bespu_gnu32.d	[yypefilter text] ▷ General ▷ Ant ▷ C(-+) ▽ Debug ⊂ Cell BE PPU SPU ▼ Environment Attached Cell Simulator Local Cell Simulator Remote Cell Simulator Remote Cell Simulator © Managed Builder Paths GNU Tools Managed Build Pa SPU Timing ▷ Install/Update ▷ Java PDT	Cell General Cell IDE Preferences SDK SYSROOT Path [/opt/cell/sysroot	په Browse Find	tine 22 Make Ta MY_ALIGN PATH_BUF_SIZE alf.h common.h libgen.h pu_my_alf_project.h stdlib.h string.h unistd.h ctx_merge_name : char[input_dtl_name : char[] kernel_name : char[]
	P Plug-in Development P Remote Tools P Run/Debug Team Team	(Restore Defaults Apply OK Cancel	!)

Tutorial finished!

This concludes the IDE tutorial!

Part 4. ALF for IDE programmer's guide

The following topics describe how to use ALF distribution template provided by IDE.

Chapter 12. ALF for IDE overview

The ALF API provides a set of functions to solve a set of parallel problems on multi-core memory hierarchy systems. This implementation of this API is focused on data parallel problems on a host-accelerator type hybrid system. ALF offers an interface to write data parallel applications without requiring architecturally dependent code. Features of ALF include data transfer management, parallel task management, double buffering, and data partitioning.

The ALF data distribution template provided by IDE, is designed to support data parallel programming style for solving data parallel problem for arrays. Similar to High Performance FORTRAN (HPF), the ALF data distribution template supports three data distribution models, which are:

- *
- CYCLIC
- BLOCK

The data distributions are presented as a serial of data transfer list entries. Data distributions provided by ALF data distribution template are for arrays of one, two, and three dimensions.

The data distributions are applied to two types of data, which are input buffer and output buffer.

Chapter 13. Data distribution overview

ALF IDE introduces data distribution directives to provide you with control over locality and data partition.

A DISTRIBUTE directive is used to indicate how data is to be partitioned among continuous memories. It specifies, for each dimension of an array, a mapping of array indices to accelerated nodes. Each dimension of an array may be distributed in one of three ways:

- * : No distribution
- BLOCK(n) : Block distribution (default: n= N/P), in which N is the size of array and P is the number of accelerated node.
- CYCLIC(n) : Cyclic distribution

BLOCK distribution

Nodes:

Let N be the number of elements in an array dimension, and let P be the number of accelerated nodes assigned to that dimension. Then, as illustrated in Figure 3, Figure 4, and Figure 5 on page 126, a BLOCK distribution divides the indices in a dimension into contiguous, equal-sized blocks of size N/P.







Figure 4. 2D array BLOCK distributions for four accelerator nodes



Figure 5. 3D array BLOCK distributions for four accelerator nodes

BLOCK(m) implies that after giving each accelerator node the array should have been distributed; if there are any elements left over an error has occurred. It implies that each accelerator node must get at most one block of elements

CYCLIC distribution

Cyclic distribution distributes elements of an array to processors in a round robin fashion. If an array, A has elements N and is mapped onto P accelerator nodes, each node gets (a maximum) total of $_{\Gamma}N / P_{1}$ separate elements. See for and as examples***WRITER COMMENT; THIS SENTENCE DOES NOT MAKE SENSE. 3D CYCLIC distribution is inefficient and confusing and is therefore NOT supported in the ALF data distribution template.



Figure 6. 1D array CYCLIC distributions for four accelerator nodes



Figure 7. 2D array CYCLIC distributions for four accelerator nodes

There are some limitation for CYCLIC distribution, they are:

- The memory address of each CYCLIC should be 16 Byte aligned.
- The size (Bytes) of each CYCLIC must be 1, 2, 4, 8 and integral times of 16 (instance: 1, 2, 4, 8, 16, 32 ...). Refer to the API definition descriptions for further information.

CYCLIC(m) retains characteristics of both BLOCK and CYCLIC distributions: in theory, blocks of m elements are grouped together which is useful for neighborhood calculations and the cyclic distribution policy should promote a reasonable degree of load balancing.

Practice with data distribution directives

The following topics describe how to work with data distribution directives:

- "A matrix addition example"
- A simple solution to solve the problem
- "Data layout and partition scheme" on page 128

A matrix addition example

This is a simple application that adds two 2D matrixes and stores the result to a third matrix. For a 2D matrix addition, the mathematical definition is as follows:

$$C = A + B, \text{ where:}$$

$$c_{ij} = a_{ij} + b_{ij}$$

or

$$\begin{vmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{vmatrix} = \begin{vmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{vmatrix} + \begin{vmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{vmatrix}$$

Understanding the problem

Simple solution

In the following analysis, assume the data to be a 1024×512 single precision floating point matrix. The following is a sample of C code that solves the problem:

```
float mat_a[1024][512];
float mat_b[1024][512];
float mat_c[1024][512];
int main(void)
{
    int i,j;
    for (i=0; i<1024; i++)
    for (j=0; j<512; j++)
    mat_c[i][j]= mat_a[i][j] + mat_b[i][j];
    return 0;
}
```

The limitation with this simple approach is that it cannot run faster on a system with many accelerators that can process the $c_{i,j} = a_{i,j} + b_{i,j}$ in parallel. There are parallel programming languages and models that can speed up the program. The following section describe steps to rewrite this program with ALF APIs for parallel acceleration.

Potential solution for parallel speed increase

In general, most matrix mathematical operations can be decomposed into similar operations on many submatrices. The operations on these submatrices can be done in parallel if there are no dependencies between them. For example, divide the matrix into 128 submatrices each of which has 64×64 elements. Then the operation can be done on each 64×64 submatrix in parallel. In theory, the computation of the 1024 x 512 matrix addition can be completed in 1/128th of the time of the simple serialized code. The following examples use the submatrix approach.

Data layout and partition scheme

Two dimensional matrices are usually represented in 2D arrays in C. The actual memory layout of the 2D array in C is in 1D arrays concatenated by the second/column index, as shown by Figure 8.



Figure 8. Memory organization of a 2D array "a[m][n]"



In the matrix addition example from the previous section, the submatrices were the basic unit of data. In the above C matrix data structure, a submatrix is part of the whole array as shown by Figure 9 and Figure 10 on page 130. This provides the following options to choose from:

- Partition Scheme A: With this partition scheme, the submatrices are a part of the whole column or row of the matrix. Or more formally, one of the submatrices of a[m][n] is defined as sa[h][v] where the h < m and v <= n. This is illustrated by the Figure 9.
- Partition Scheme B: With this partition scheme, we define the submatrices as a set of adjacent full length rows of the matrix. One of the submatrices of a [m] [n] is defined as "sa[h] [v]" where the h < m and v == n. See Figure 10 on page 130



Figure 9. Partition scheme A: Data partition of 2D sub matrix



Figure 10. Partition Scheme B: Data partition of 2D sub matrix

Figure 7 on page 127 shows the differences between the two schemes. The data of the submatrix in scheme A is collected from disjointed segments in the data buffer of the matrix. For scheme B, the submatrix is from one contiguous segment of the matrix. Mathematically this makes no significant difference, but the data movement in our matrix addition example is significantly more complex in scheme A than in scheme B, as can be seen by the following example pseudo code with data distribution directives.

```
DECLARE ACCELERATE_NODES = 6;
```

```
float mat_a[1024][512];
float mat_b[1024][512];
float mat_c[1024][512];
float mat_c[1024][512];
CYCLIC(512,8) ONTO mat_a
CYCLIC(512,8) ONTO mat_b
CYCLIC(512,8) ONTO mat_c
for (i=0; i<512; ++i)
for (j=0; j<8; ++j)
mat_c[i][j]= mat_a[i][j] + mat_b[i][j];
```

Based on the above analysis, scheme B is preferred in this matrix addition example. Remember that this situation can change in some real world scenarios where large contiguous data movement might not be supported.

Chapter 14. ALF IDE wizard overview

This section describes the ALF IDE wizard.

It covers the following topics:

- "Basic structure of an ALF application"
- "What does the ALF IDE wizard do?"

Basic structure of an ALF application

The basic structure of an ALF application is shown in Figure 11. In general the flow is linear.

On the host side, you first initialize the ALF runtime and then create a compute task. After you have created the task, you add work blocks to the work queue of the task. Then, you wait for the task to complete and shut down the ALF runtime to free up allocated resources.

On the accelerator side, after an instance of the task is spawned, it waits for pending work blocks to be added to the work queue. Then the **alf_comp_kernel** is called for each work block. If the partition location attribute of a task is partitioned in accelerated node, **alf_accel_intput_list_prepare** is called before the compute kernel is invoked and **alf_accel_output_list_prepare** is called after the compute kernel exits.



Figure 11. ALF application structure and execution flow

What does the ALF IDE wizard do?

The ALF IDE wizard aims to release you from trivial ALF programming tasks, and to help you focus on your problem logic.

The ALF IDE wizard automatically generates source code for host and accelerator code. In most cases, you only needs to add code for computing kernel and data initialization.

The ALF IDE wizard does several tasks works automatically, as described in the following figure.



Figure 12. Tasks performed by the ALF IDE wizard

Figure 12 shows that the ALF IDE wizard reduces some trivial and routine tasks, such as initialization, and preparation of the input/output transfer list. The ALF IDE wizard also generates the correct Makefiles for you.

The ALF IDE Wizard does not automatically generate source code for the computing kernel and data initialization tasks. You need to generate code for your computing kernel and data initialization.

IDE also cannot handle the data transfer list for irregular data movement. In this case, it is suggested that you use the initialization and skeleton template for the ALF project to generate an empty ALF project, and code your own application.

If you need to add any other libraries or headers into application, you need to revise the project's properties to add these dependencies.

Chapter 15. Programming tips using the ALF IDE wizard

The following topics provide tips about how to program using the ALF IDE wizard.

Data layout and partition scheme

Data partition is crucial to the ALF programming model. Incorrect data partitioning and data layout design either prevents ALF from being applicable or results in degraded performance. For the best performance, use the following guidelines for data partitioning and data layout design.

Note: Data partition design and layout can be platform-dependent, so the best design for one architecture might not perform well on another.

Data partition and layout is closely coupled with compute kernel design and implementation, so you should them considered simultaneously.

• Use the correct data partition size

Often, the accelerator local memory or data cache is very limited. A significant performance penalty can occur if the partitioned data cannot fit into this memory limitation. For example, on Cell/B.E. architecture, if a single block of partitioned data is larger than 128 KB, it might not be possible to support double buffering on the SPE side. This can result in up to 50% performance loss.

• Minimize the number of data movements.

A large amount of data movement can cause performance loss in applications. Improve performance by avoiding unnecessary data movements. Use the correct data distribution size to improve the performance.

• Know address alignment limitations on specific platforms

If data is stored on aligned addresses, the processor can access it faster than data on unaligned addresses. On platforms such as Cell/B.E., the data movement from the local memory of the SPE can only be based on 16 byte aligned addresses.

For more detail about data partitioning restrictions, refer to "Data partitioning limitations" on page 143.

ALF host program and data transfer lists

One important decision for data transfer list is to choose whether to use accelerator side data transfer list generation. Because there might be a large number of accelerators used in one compute task if the data transfer list is somewhat complex, the host might not be able to generate work blocks faster than the accelerators can process. In this case, you can supply the data needed for data transfer list generation in the parameters of the work block and use the accelerators to generate the data transfer lists based on these parameters.

You can also start a new task for queuing while another task is running. You can then prepare the work blocks for the new task in advance before the task actually runs. ALF IDE wizard generates data transfer lists both in host and in accelerator simultaneously. You can use the macro ACCELERATOR_PARTITION defined in common.h to switch between them.

Chapter 16. Programming with ALF IDE wizard

This section describes how to program with the ALF IDE wizard.

It covers the following topics:

- "Using the ALF IDE wizard to create an ALF project" on page 136
- "ALF IDE wizard parameters" on page 137
- "Using user-defined types" on page 140
- "Example: Matrix addition" on page 140

Using the ALF IDE wizard to create an ALF project

This topic describes what you need to do to create an ALF project.

You need to follow the following steps to generate an ALF project.

- 1. Create an ALF project and give the project a name.
- 2. Specify the expected stack size.
- 3. Select the expected number of accelerators (1, 2, 16, and 0: for all available).
- 4. Select the partition method (host or accelerator).
- 5. Describe the data distribution model for each buffer. For each new buffer input, performing the following steps:
 - a. Enter the variable name, element type (the type for each element in the buffer), and basic unit (an ALF predefined type or the same as element type) of the element
 - b. Specify the buffer type (INPUT or OUTPUT)
 - **c**. Specify the dimension and size (1, ... dimension) for each dimensions of the array
 - d. Specify the distribution model for this array (CYCLIC or BLOCK), and parameters for this distribution model
- 6. Finish. The ALF IDE wizard generates code according to your input.
- 7. Revise the generated code as required.


Figure 13. Creating ALF project

ALF IDE wizard parameters

The following table lists the user input parameters.

Table 2.	Basic	user	input	parameters
10010 21	Duolo	4001	inpac	paramotoro

Token	Description	Data range
<pre>\$PROJECT_NAME</pre>	Project name (Step 1)	Strings
\$STACK_SIZE	Expected stack size (Step 2)	From 10 Bytes to 246 KBytes
\$EXP_ACCEL_NUM	Expected number of accelerator (Step 3)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, and 0 for all available. Note: All available option is invalid when any buffer's data distribution model is BLOCK
<pre>\$PARTITION_METHOD</pre>	Partition model, host, or accelerator (Step 4)	0 : Partition by host 1 : Partition by accelerator
\$VARIABLE_NAME	Name of the data array (Step 5a)	Valid variable name in C language, it is case-sensitive. Different buffer should have different variable name.
\$ELEMENT_TYPE	Type of element for the data array indicated by \$VARIABLE_NAME (Step 5a)	C language built in type or user defined data type.

Table 2. Basic user input parameters (continued)

Token	Description	Data range
\$ELEMENT_UNIT	Array data type. (Step 5a)	Basic data types. The choices are:ALF data types. There are 6 types:
		ALF_DATA_INT16 ALF_DATA_INT32 ALF_DATA_INT64 ALF_DATA_FLOAT ALF_DATA_DOUBLE ALF_DATA_BYTE
		• As defined by \$ELEMENT_TYPE
\$BUFFER_TYPE	Buffer type of the data array indicated by \$VARIABLE_NAME. Input buffer or output buffer. (Step 5b)	ALF_BUFFER_INPUT: input buffer ALF_BUFFER_OUTPUT: output buffer
\$NUM_DIMENSION	Number of dimension for array indicated by \$VARIABLE_NAME. (Step 5c)	1, 2, or 3
\$DIMENSION_SIZE_X	X dimension size for array indicated by \$VARIABLE_NAME.	1,
	For a 3D dimension array A, its dimensions represented as A[Z][Y][X]	
	For a 2D dimension array B, its dimensions represented as B[Y][X]	
	For a 1D dimension array C, its dimensions represented as C[X](Step 5c)	
\$DIMENSION_SIZE_Y	Y dimension size for array indicated by \$VARIABLE_NAME. Only available when the array dimension is 2 or 3.	1, 0 when the dimension of array is less than 2.
	For a 3D dimension array A, its dimensions represented as A[Z][Y][X]	
	For a 2D dimension array B, its dimensions represented as B[Y][X](Step 5c)	
\$DIMENSION_SIZE_Z	Z dimension size for array indicated by \$VARIABLE_NAME. Only available when the array dimension is 3.	 0 when the dimension of array is less than 3.
	For a 3D dimension array A, its dimensions represented as A[Z][Y][X].(Step 5c)	
\$DISTRIBUTION_MODEL_X	The distribution model for X dimension of data array which indicated by \$VARIABLE_ (Step 5d)	BLOCK, CYCLIC, or *

Table 2. Basic user input parameters (continued)

Token	Description	Data range
\$DISTRIBUTION_SIZE_X	The distribution size for X dimension distribution (indicated by \$VARIABLE_NAME) for data array (indicated by \$VARIABLE_NAME). (Step 5d)	CYCLIC: User input integer number, and limited into (1 - \$DIMENSION_SIZE_X). For BLOCK distribution, it is calculated by IDE as: \$DISTRIBUTION_SIZE_X = \$DIMENSION_SIZE_X / \$EXP_ACCEL_NUM *: \$DISTRIBUTION_SIZE_X = \$DIMENSION_SIZE_X
<pre>\$DISTRIBUTION_MODEL_Y</pre>	The distribution model for Y dimension of data array which indicated by \$VARIABLE_NAME Only available when the dimension of data array is 2 or 3.	BLOCK, CYCLIC, or Should not appear in the parameter file when the dimension of array is less than 2.
\$DISTRIBUTION_SIZE_Y	The distribution size for Y dimension distribution (indicated by \$DISTRIBUTION_MODEL_Y) for data array (indicated by \$VARIABLE_NAME). Only available when the dimension of data array is 2 or 3.(Step 5d)	CYCLIC: User input integer number, and limited into (1 - \$DIMENSION_SIZE_Y). For BLOCK distribution, it is calculated by IDE as: \$DISTRIBUTION_SIZE_Y = \$DIMENSION_SIZE_Y / \$EXP_ACCEL_NUM *: \$DISTRIBUTION_SIZE_Y = \$DIMENSION_SIZE_Y Should not appear in the parameter file when the dimension of array is less than 2.
<pre>\$DISTRIBUTION_MODEL_Z</pre>	The distribution model for Z dimension of data array which indicated by \$VARIABLE_NAME Only available when the dimension of data array is 3.(Step 5d)	BLOCK or * Should not appear in the parameter file when the dimension of array is less than 3.
\$DISTRIBUTION_SIZE_Z	The distribution size for Z dimension distribution (indicated by \$DISTRIBUTION_MODEL_Z) for data array (indicated by \$VARIABLE_NAME). Only available when the dimension of data array is 3.(Step 5d)	BLOCK: \$DISTRIBUTION_SIZE_Z = \$DIMENSION_SIZE_Z / \$EXP_ACCEL_NUM *: \$DISTRIBUTION_SIZE_Z = \$DIMENSION_SIZE_Z Should not appear in the parameter file when the dimension of array is less than 3.

Data partition limitations

There are some limitations for data partitions.

The partition is eventually turned into a serial of data transfer entry. The number of these entries for each buffer after partitioning should be equal. The following examples illustrate some valid and invalid data distributions for arrays: A[1024][512], B[512][128].

- (CYCLIC(512), CYCLIC(16)) ONTO A, and (CYCLIC(128), CYCLIC(8) ONTO B is valid, because the number of DT entries for these two distributions are equal. (For A, the number of DT entry is (1024/512) * (512 / 16), and for B, the number of DT entry is (512 / 128) * (128/6))
- (*, CYCLIC(16)) ONTO A, and (CYCLIC(128), CYCLIC(16)) ONTO B is valid, because the numbers of DT entry for these two distributions are equal, it is 32.

 (CYCLIC(512), CYCLIC(16)) ONTO A, and (CYCLIC(128), CYCLIC(16)) is invalid, because the numbers of DT entry for these two distributions are not equal.

The data size of each data transfer entry must be a multiple of 16 Byte (16 Bytes, 32 Bytes, and so on).

The sum of total size of buffer and free stack size should not exceed the size of Local Storage (256 KBytes) for Cell/B.E. platform.

Using user-defined types

The ALF IDE wizard supports user-defined types. When your input is not a C-language built-in type, the ALF IDE wizard generates a user-defined type in common.h.

For example, when you input an array as my_structure a[1024], a struct named my_structure is presented in common.h.

```
typedef
struct _my_structure_t
{
//TODO: Code your data structure here, make sure the size (Byte)
//of this structure is 1, 2, 4, 8 or power of 16.
char
pad[16];
}my_structure;
```

You can revise the structure generated by IDE as you require. You need to make sure that the size of the structure is 1, 2, 4, 8, or power of 16 (Byte) as this is a limitation of the Cell/B.E. platform.

The following types are treated as C programming language built-in types.

IDE supported C programming language built-in types

```
"char", "unsigned char", "signed char", "short", "unsigned short",
"int", "unsigned int","long", "signed", "unsigned","unsigned long",
"long long","unsigned long long", "float", "double", "long double",
"_Bool", "float _Complex", "double _Complex", "long double _Complex"
```

Example: Matrix addition

The following topics describe how to code your computing kernel.

Step-by-step instructions

Refer to the Chapter 11, "IDE tutorial," on page 31 for a step by step walkthrough of how to use the ALF IDE wizard.

Coding your computing kernel

You must code your computing kernel and data initialization. In this step, you code the computing kernel in accelerated code.

By editing spu_matrix_add.c, enter the following code into **alf_accel_comp_kernel**. The computing kernel will look like the following example (user-defined code is marked as bold).

Computing kernel for matrix addition

```
int alf accel comp kernel(void *p task context attribute
((unused)),void *p_parm_ctx_buffer,void *p_input_buffer,void
*p_output_buffer,void *p_inout_buffer __attribute__ ((unused)),unsigned
int current_count __attribute__ ((unused)),unsigned
int total_count __attribute__ ((unused)))
{
 float *mat_a;
 float *mat_b;
 float *mat c;
 my_param_t *p_param = (my_param_t *) p_parm_ctx_buffer;
 mat a = (float*) ((char*)p input buffer + p param->c.buffer offset[0]);
mat_b = (float*) ((char*)p_input_buffer + p_param->c.buffer_offset[1]);
 mat c = (float*) ((char*)p_output_buffer + p_param->c.buffer_offset[2]);
 //TODO: Code your computing kernel here
 int i;
 for(i=0; i<p_param->c.size[0]; ++i)
       ł
        mat_c[i] = mat_a[i] + mat_b[i];
       }
 return 0;
}
```

We will also add a data initialization code in ppu_matrix_add.c, and call it in main (). As illustrated in the next sample and marked as bold.

```
Data preparation in PPU side
```

```
static void prepare_data()
{
int i, j;
for (i=0; i<1024; ++i)
for (j=0; j<512; ++j)
{
mat_a[i][j] = i * 2.0 + j;
mat b[i][j] = i + 3.0 + j;
mat_c[i][j] = 0.0;}
}
int main()
prepare_data();
alf_init(NULL, &alf_handle);
if ((err = alf query system info(alf handle, ALF QUERY NUM ACCEL, &nodes)) < 0)
fprintf(stderr, "Error in query system information\n");
return (-1);
}
. . .
}
```

Chapter 17. Platform-specific constraints for ALF IDE Wizard on Cell/B.E. architecture

The section describes the following platform-specific constraints:

- "Data distribution limitations"
- "Data partitioning limitations"
- "Local memory constraints for SPE" on page 144
- "User-defined type size limitations" on page 146

Data distribution limitations

This topic describes the limitations for the various ALF data distribution policies.

For BLOCK distribution, the expected accelerator number should be exact, all available is not allowed. This means if there has any buffer for which the distribution model is BLOCK, the option all available options cannot be selected.

CYCLIC distribution is only available when the dimension of an array is one or two.

Data partitioning limitations

For each array, the number of blocks after partitioning should be equal, as they are put together into the same ALF work block. Errors occur when the number of partitioned blocks is not equal.

The size of each block should be bound into 16 bytes (such as 16 byte, 32 byte, 48 byte, and so on). The following describes how to calculate the size of data block after partitioning.

- data_transfer_size = sizeof(\$ELEMENT_TYPE) * \$DISTRIBUTION_SIZE_X, when the array dimension is one
- data_transfer_size = sizeof(\$ELEMENT_TYPE) * \$DISTRIBUTION_SIZE_X * \$DISTRIBUTION_SIZE_Y, when the array dimension is two
- data_transfer_size = sizeof(\$ELEMENT_TYPE) * \$DISTRIBUTION_SIZE_X * \$DISTRIBUTION_SIZE_Y * \$DISTRIBUTION_SIZE_Z, when the array dimension is three

If ((data_stransfer_size % 16) != 0), an error occurs.

To sum up, for ALF forCell/B.E., be aware of the following data transfer list constraints:

- Data transfer information for a single working block can consist of up to eight data transfer lists for each direction transfer (local store to main memory and vice versa)
- Each data transfer list consists of up to 2048 data transfer entries
- Each entry can describe up to 16 KB of data transfer between the continuous area in main memory and local storage
- The local store area described by each entry within the same data transfer list must be contiguous

• The transfer size and effective address low 32 bits for each data transfer entry must be 16 bytes aligned

Local memory constraints for SPE

The size of local memory on the accelerator is 256 KB and is shared by code and data. Memory is not virtualized and is not protected. A typical memory map of an SPU program is given in Figure 14 on page 145. There is a runtime stack above the global data memory section. The stack grows from the higher address to the lower address until it reaches the global data section. Due to the limitation of programming languages and compiler/linker tools, you cannot predict the maximum stack usage when you develop the application and when the application is loaded. If the stack requires larger storage space than was allocated, there will be a stack overflow exception. The current approach is to dedicate all the free memory to the stack. New compiler tools also support the option to generate stack boundary check code according to the process given in the SPU ABI document. The SPU application will be shutdown when there is a stack overflow and a message is sent to the PPE.

ALF allocates the work block buffers directly from the memory region higher than the runtime stack. This is implemented by moving the stack pointer (or by pushing a large amount of data into the stack). For ALF, the larger the buffer is, the better it can optimize the performance of a task by using techniques such as double buffering. It is better to let ALF allocate as much as possible memory from the stack. Because of this, you must estimate the maximum stack usage information for the user code plus ALF and programming language runtime code. Estimate the stack usage and input the value into IDE. If the stack size is too small at runtime, a stack overflow occurs and it causes unexpected exceptions such as wrong results or a program crash. Be sure to leave enough local memory space for runtime stack usage in the early stage of your coding and data partition designs. Reduce the stack size to make more memory for ALF runtime usage only when the actual requirement can be estimated more accurately.



Figure 14. SPU local memory map - common Cell/B.E. application



Figure 15. SPU local memory map - common ALF application

Because of SPE space limitations, the sum of data size partitioned into each accelerator node and the stack size should not exceed 246 KBytes (The SPE space is 256 KB). The data size for each accelerator can be calculated as follows (for the size of data transfer list, refer to "Data partitioning limitations" on page 143):

```
sum = 0;
for_each(IO_BUFFER)
{
    sum +=
    data_transfer_size;
```

```
}
sum +=
STACK_SIZE;
if (sum >246 * 1024) return error;
```

IDE reports an error when this limitation is exceeded.

User-defined type size limitations

The size of a user-defined type should be a multiple of 16 Bytes. IDE does not check this.

A tip for generating a 16 byte boundary type, is to declare data structure as follows, and the compiler will generate a data structure with a size that is bounded into 16 bytes:

```
User-defined type bounded into 16 bytes
typedef struct _my_type_t
{
    struct my_content_t
    {
        //Your parameters
     }c;
     unsigned char _pad[(sizeof(struct _my_type_t) + 15) & 0xFF10];
}
my_type_t;
```

Part 5. Appendixes

Appendix A. Related documentation

This topic helps you find related information.

Document location

Links to documentation for the SDK are provided on the IBM developerWorks Web site located at:

http://www.ibm.com/developerworks/power/cell/

Click the **Docs** tab.

The following documents are available, organized by category:

Architecture

- Cell Broadband Engine Architecture
- Cell Broadband Engine Registers
- SPU Instruction Set Architecture

Standards

- C/C++ Language Extensions for Cell Broadband Engine Architecture
- Cell Broadband Engine Linux Reference Implementation Application Binary Interface Specification
- SIMD Math Library Specification for Cell Broadband Engine Architecture
- SPU Application Binary Interface Specification
- SPU Assembly Language Specification

Programming

- Cell Broadband Engine Programmer's Guide
- Cell Broadband Engine Programming Handbook
- Cell Broadband Engine Programming Tutorial

Library

- Accelerated Library Framework for Cell Broadband Engine Programmer's Guide and API Reference
- Basic Linear Algebra Subprograms Programmer's Guide and API Reference
- Data Communication and Synchronization for Cell Broadband Engine Programmer's Guide and API Reference
- Example Library API Reference
- Fast Fourier Transform Library Programmer's Guide and API Reference
- LAPACK (Linear Algebra Package) Programmer's Guide and API Reference
- Mathematical Acceleration Subsystem (MASS)
- Monte Carlo Library Programmer's Guide and API Reference
- SDK 3.0 SIMD Math Library API Reference
- SPE Runtime Management Library
- SPE Runtime Management Library Version 1 to Version 2 Migration Guide
- SPU Runtime Extensions Library Programmer's Guide and API Reference

• Three dimensional FFT Prototype Library Programmer's Guide and API Reference

Installation

• SDK for Multicore Acceleration Version 3.1 Installation Guide

Tools

- Getting Started XL C/C++ for Multicore Acceleration for Linux
- Compiler Reference XL C/C++ for Multicore Acceleration for Linux
- Language Reference XL C/C++ for Multicore Acceleration for Linux
- Programming Guide XL C/C++ for Multicore Acceleration for Linux
- Installation Guide XL C/C++ for Multicore Acceleration for Linux
- Getting Started XL Fortran for Multicore Acceleration for Linux
- Compiler Reference XL Fortran for Multicore Acceleration for Linux
- Language Reference XL Fortran for Multicore Acceleration for Linux
- Optimization and Programming Guide XL Fortran for Multicore Acceleration for Linux
- Installation Guide XL Fortran for Multicore Acceleration for Linux
- Performance Analysis with the IBM Full-System Simulator
- IBM Full-System Simulator User's Guide
- IBM Visual Performance Analyzer User's Guide

IBM PowerPC® Base

- *IBM PowerPC Architecture*TM Book
 - Book I: PowerPC User Instruction Set Architecture
 - Book II: PowerPC Virtual Environment Architecture
 - Book III: PowerPC Operating Environment Architecture
- IBM PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Environments Manual

Appendix B. Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

The following list includes the major accessibility features:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- · Keys that are tactilely discernible and do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

IBM and accessibility

See the IBM Accessibility Center at http://www.ibm.com/able/ for more information about the commitment that IBM has to accessibility.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing 2-31 Roppongi 3-chome, Minato-ku Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation Software Interoperability Coordinator, Department 49XA 3605 Highway 52 N Rochester, MN 55901 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com[®] are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol ([®] or [™]), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

Adobe[®], Acrobat, Portable Document Format (PDF), and PostScript[®] are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc., in the United States, other countries, or both and is used under license therefrom.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of the manufacturer.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of the manufacturer.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any data, software or other intellectual property contained therein.

The manufacturer reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by the manufacturer, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

THE MANUFACTURER MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THESE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

Α

ACCELERATOR_PARTITION 133 address alignment limitations in ALF 133 ALF buffer 123 computing kernel 140 data distribution 125 data distribution template 123 data layout 133 data partitioning limitations 143 matrix addition sample 127 parameters 137 partitioning 133 programming tips 133 type size limitations 146 user-defined type 140 wizard for IDE 123 ALF IDE installing templates 7 ALF IDE wizard 131 using 136 alf_accel_intput_list_prepare 131 alf_accel_output_list_prepare 131 applications running Cell/B.E. 17 architecture supported by IDE 5 array in ALF 125 attached Cell/B.E. simulator 15 authentication configuring 25

В

BLOCK limitations 143 BLOCK disribution 125 buffer 123 builder path 17, 21

С

C Development Tools see CDT 3 C/C++ cell target application 17 creating local application 17 CDT 3 XLC error parser 23 Cell/B.E. box 15 compiler error parser 23 computing kernel 140 configuring authentication 25 debugger 17 path 17 CYCLIC distribution 125

D

data distribution (ALF) 125 distribution limitations 143 layout 133 partition limitations 143 transfer list 133 wizard 133 debugger configuring 17 debugging Cell/B.E. applications 17 creating environment 15 documentation v, 149 SDK v

Ε

Eclipse installing 7 supported version 5 uninstalling 11 environment attached Cell/B.E. simulator 15 Cell/B.E. box 15 local Cell/B.E. simulator 15 remote Cell/B.E. simulator 15 error parser 23

F

FORTRAN 123

G

gdbserver 27 GNU tool path for Cell/B.E. systems 17 path for PPC systems 21 GNU tool chain 3

IBM Full System Simulator see simulator 3 IDE definition 3 limitations 27 wizard for ALF 131 XLC error parser 23 installing ALF IDE templates 7 IDE 7 JRE 7

J

Java supported version 5 USE_JAVA_API option 27 JRE installing 7 JVM installing 7

K

key 25 known issues 27

L

limitations ALF data distribution 143 ALF data partitioning 143 ALF type size 146 BLOCK distribution 143 GDB server 27 Java API 27 localhost 27 MAC address 27 remote launch directory 27 Linux PATH variable 7 localhost 27

Μ

MAC address 27

0

OpenSSH 25 operating system 5

Ρ

parameters ALF 137 password 25 path configuring 17, 21 PATH variable 7 private key 25 public key 25

R

remote Cell/B.E. simulator 15 remote launch directory 27 requirements architecture 5 Java 5 operating system 5 requirements (continued) software 5

S

sample matrix addition in ALF 140 SDK documentation 149 Secure Shell see SSH) 25 simulator 3 software supported by IDE 5 SPE limitations SPE local memory 144 memory constraints 144 SSH 25

Т

troubleshooting 27 type user-defined 140 user-defined limitations 146

U

uninstalling ALF IDE templates 11 IDE 11 USE_JAVA_API option 27

W

work-arounds 27

X

XL tool path for Cell/B.E. systems 17 path for PPC systems 21



Printed in USA

SC34-2561-00

