JN5148 Software Developer's Kit Installation and User Guide

JN-UG-3064
Revision 2.0
22 November 2010

Contents

Part III: Appendices

About this Manual

This manual provides guidance on installing and using the Software Developer's Kit (SDK) for the NXP JN5148 microcontroller, targeted at wireless network applications. The Eclipse Integrated Development Environment (IDE) is provided as a component of the JN5148 SDK and part of this manual is devoted to using Eclipse in the development of applications for the JN5148 device.

Note: This manual incorporates information from the former *Eclipse IDE User Guide (JN-UG-3063)*.

Organisation

The manual is divided into three parts:

- Part I: Introduction and Installation comprises three chapters:
  - Chapter 1 introduces the JN5148 SDK, including its contents and the wireless network protocols that it supports.
  - Chapter 2 describes how to install the JN5148 SDK Toolchain.
  - Chapter 3 describes how to install the JN5148 SDK Libraries.
- Part II: Eclipse Integrated Development Environment comprises four chapters:
  - Chapter 4 introduces the Eclipse platform and describes how to install NXP external components into Eclipse.
  - Chapter 5 describes how to create a project in Eclipse and build an application to be run on the JN5148 device.
  - Chapter 6 describes how to download a built application to the Flash memory of a JN5148-based module or board.
  - Chapter 7 describes how to debug an application running on a JN5148 device.
- Part III: Appendices comprises four appendices providing useful procedures that may be required during installation or use of the SDK.

Conventions

Files, folders, functions and parameter types are represented in **bold** type.

Function parameters are represented in *italics* type.

Code fragments are represented in the `Courier New` typeface.

This is a **Tip**. It indicates useful or practical information.

This is a **Note**. It highlights important additional information.

*This is a **Caution**. It warns of situations that may result in equipment malfunction or damage.*

Acronyms and Abbreviations

API	Application Programming Interface
CLI	Command Line Interface
GDB	GNU debugger
GUI	Graphical User Interface
IDE	Integrated Development Environment
ISR	Interrupt Service Routine
JenNet	Jennic Network
JenOS	Jennic Operating System
LPRF	Low-Power Radio Frequency
MAC	Media Access Control
SDK	Software Developer's Kit
ZPS	ZigBee PRO Stack

Related Documents

JN-UG-3007 JN51xx Flash Programmer User Guide

JN-UG-3024 IEEE 802.15.4 Stack User Guide

JN-UG-3041 JenNet Stack User Guide

JN-UG-3048 ZigBee PRO Stack User Guide

JN-UG-3075 JenOS User Guide

Feedback Address

If you wish to comment on this manual, please provide your feedback by writing to us (quoting the manual reference number and version) at the following postal address or e-mail address:

Applications
NXP Laboratories UK Ltd
Furnival Street
Sheffield S1 4QT
United Kingdom

doc@jennic.com

Part I:
Introduction and Installation

1. JN5148 SDK Overview

The JN5148 Software Developer's Kit (SDK) is designed to aid software development for the NXP JN5148 microcontroller, targeted at wireless network applications based on protocols such as ZigBee PRO, JenNet and IEEE 802.15.4. This chapter introduces the SDK and the software development options that it provides, before the SDK installation instructions are given in Chapter 2 and Chapter 3.

***Caution**: Before installing the JN5148 SDK described in this manual, you must remove any previous SDK version other than the JN5139 SDK - see* Appendix D.

The JN5148 SDK supports application development within the Eclipse Integrated Development Environment (IDE), which is provided as part of the SDK. Guidance on the use of Eclipse is provided in Part II: Eclipse Integrated Development Environment of this manual.

1.1 SDK Contents

The JN5148 SDK is supplied as two independent installers:

- **JN5148 SDK Toolchain (JN-SW-4041)**: This installs the NXP software tools that you will use to prepare your wireless network applications. These utilities include development, compiler and Flash programming tools. For more details, refer to Section 1.1.1.
- **JN5148 SDK Libraries (JN-SW-4040):** This installs the NXP software libraries that will help streamline your wireless network application development. These libraries include Application Programming Interfaces (APIs) for the ZigBee PRO, JenNet and IEEE 802.15.4 protocol stacks, as well as JenOS (Jennic Operating System). For more details, refer to Section 1.1.2.

To complete the set-up of your application development environment, you must also install plug-ins which provide configuration editors and other external tools in the Eclipse IDE. Installation instructions for these components are provided in Chapter 2 and Chapter 3.

The above SDK installers are available from **www.nxp.com/jennic**.

1.1.1 SDK Toolchain Installer

The JN5148 SDK Toolchain is supplied in the file **JN-SW-4041-SDK-Toolchain-v*X.Y*.exe**. This includes the following development tools:

- Cygwin CLI (Command Line Interface)
- Eclipse IDE (Integrated Development Environment)
- Debugging tools (standalone or as Eclipse plug-ins)
- JN51xx Compiler Tools
- JN51xx Flash Programmer (standalone or as Eclipse plug-ins)

You should normally install all of these components. The JN5148 SDK Toolchain installation procedure is provided in Chapter 2. In addition to the above tools, OS and network configuration editors are available for ZigBee PRO as Eclipse plug-ins and, if required, must be installed as described in Chapter 4.

Note: Application development for the JN5148 device is intended to be conducted completely within the Eclipse IDE. It is, however, possible to develop your application code using another editor and to build the application using makefiles.

1.1.2 SDK Libraries Installer

The SDK Libraries are supplied in the file **JN-SW-4040-SDK-Libraries-v*X.Y*.exe**. This includes a number of APIs containing C functions, as well as other software components:

- ZigBee PRO networking layer and APIs
- JenNet networking layer and APIs
- IEEE 802.15.4 networking layer and API
- JenOS APIs
- JN51xx Integrated Peripherals API
- LPRF Board API
- Configuration tool command line utilities

All of the above components are installed. The libraries installation procedure is provided in Chapter 3 (but you must first install the toolchain - see Chapter 2).

Note: Application access to the JN5148 on-chip peripherals is provided by the Integrated Peripherals API. In addition, JN5148-EK010 evaluation kit board resources can be accessed using the LPRF Board API.

1.2 Wireless Network Protocol Options

The JN5148 SDK offers a choice of three wireless network protocols:

- **IEEE 802.15.4:** This is an industry-standard protocol which provides the low-level functionality for implementing wireless network communications - for example, it provides an interface with the transmission medium (i.e. radio). The JenNet and ZigBee PRO protocols are built on top of IEEE 802.15.4, but an application can also be designed to interface directly with the IEEE 802.15.4 stack layers and an API is provided to facilitate this interaction.
- **JenNet:** This is a proprietary protocol (Jennic Network) which is built on IEEE 802.15.4 to simplify wireless network application development by providing a Network stack layer. JenNet is supplied with an API, known as the Jenie API, to facilitate the interaction between the application and the JenNet stack.
- **ZigBee PRO:** This is an industry-standard protocol which is built on IEEE 802.15.4 to simplify wireless network application development by providing a Network stack layer that supports Mesh networking. APIs are provided to facilitate the interaction between the application and the ZigBee PRO stack. The ZigBee PRO APIs must be used in conjunction with the APIs of JenOS (Jennic Operating System).

A comparison of these three protocols is presented in the table below.

Criteria	IEEE 802.15.4	JenNet	ZigBee PRO
Recommended Topologies	Star Point-to-point	Tree Star Linear	Mesh
Maximum Network Size	50 nodes	500 nodes	100 nodes
Network Recovery	None	Self-repairing	Self-repairing
Development Complexity	Medium	Low	High
Available Application Code Space *	Co-ord/Router: 115 Kbytes End Device: 115 Kbytes	Co-ord/Router: 85 Kbytes End Device: 95 Kbytes	Co-ord/Router: 36 Kbytes End Device: 48 Kbytes
Standards Compliance	IEEE 802.15.4 standard	Proprietary networking layer built on standard IEEE 802.15.4 layers	ZigBee standard networking layer built on standard IEEE 802.15.4 layers
Third-party Interoperability	No provision	No provision	Interoperability through ZigBee public profiles and compliance/certification
Licensing Costs	Free	Free	ZigBee Alliance membership and product certification fees
Solutions from NXP	Cable Replacement Remote Control	Cable Replacement Active RFID Intelligent Lighting	Smart Energy Home Automation

Table 1: Protocol Stack Selection Criteria

* Based on the appropriate NXP application template

1.3 Installation Pre-requisites

This section details the pre-requisites for your wireless network application development.

Before installing the JN5148 SDK, make sure you have the following:

- A machine with the following specification:
  - Windows Vista, XP or 2000 operating system
  - At least 240 MB of hard disk space available
- Administrator rights on the machine
- The following SDK installers (available from **www.nxp.com/jennic**):
  - **JN-SW-4041-SDK-Toolchain-v*X.Y*.exe**
  - **JN-SW-4040-SDK-Libraries-v*X.Y*.exe**

> **Note**: If the JN5139 SDK (installers JN-SW-4030 and JN-SW-4031) is already installed then you can choose to uninstall it or not. However, **any other** previous SDK installation must be removed before installing the JN5148 SDK - see Appendix D.

1.4 Software Updates

Once you have installed the SDK, you can check for software updates at any time. To do this, a facility is provided to check the NXP web site for the latest software and to download a new software version, if it exists. This compares the installed SDK version against the latest SDK version on the web site.

To start a software check/update, in the Windows **Start** menu follow the path:

Start > All Programs > Jennic > Check for updates

The software re-installation (if any) will be performed automatically. Simply follow the on-screen instructions.

2. Installing the SDK Toolchain

This chapter describes how to install the Toolchain part of the JN5148 SDK, which must be installed before the Libraries part (see Chapter 3). The JN5148 SDK Toolchain installer (JN-SW-4041) is supplied as the following file:

JN-SW-4041-SDK-Toolchain-v*X.Y*.exe

This file is available from **www.nxp.com/jennic**.

Note: If the JN5139 SDK (installers JN-SW-4030 and JN-SW-4031) is already installed then you can choose to uninstall it or not. However, **any other** previous JN51xx SDK installation must be removed before installing the JN5148 SDK - see Appendix D.

Note: If you are currently using another version of the SDK, you are recommended to back up your **SDK/ Application** folder before installing the JN5148 SDK.

Caution: *Do not install the JN5148 SDK Libraries until you have installed the JN5148 SDK Toolchain, as described in this chapter.*

2.1 Toolchain Contents

The software components that can be installed from the JN5148 SDK Toolchain are listed in the table below.

Component	Description
Cygwin	This is the Cygwin Command Line Interface (CLI) which emulates Linux. The SDK contains an NXP edition of Cygwin with reduced functionality. You can use this as a standalone development environment, if you wish, but it is also needed for the JN51xx compiler tools and for Eclipse. You must install this component, unless you already have a full Cygwin installation on your machine. Also refer to the important Cygwin information below this table.
Eclipse	This is the graphical Integrated Development Environment (IDE) used to develop applications for the JN5148 device. For more information on Eclipse, refer to Part II: Eclipse Integrated Development Environment of this manual.
Flash Programmer	This is the JN51xx Flash Programmer that you will need to download your built applications to the Flash memory used by the JN5148 device. You will always need this component. The Flash programmer is available from within the Eclipse environment.
Debugging tool	A tool is provided for debugging applications. This is supplied as an Eclipse plug-in, so is available from within the Eclipse environment. For more information, refer to Chapter 7.
JN51xx Compiler Tools	These tools include the JN51xx compiler and linker, which are always needed. The tools will be installed into the **Tools** directory within the **Jennic** installation folder and can be called from the Cygwin command line or from within Eclipse.

Table 2: Toolchain Components

It is important to note the following in relation to the installation of Cygwin:

- If you already have a **full** Cygwin environment on your machine, there is no need to install the NXP Cygwin environment from the SDK and you are advised **not** to do so, as the new Cygwin installation will overwrite the registry settings of the previous installation.
- If you intend to keep an earlier JN5139 SDK Toolchain installation (JN-SW-4031) that includes the Jennic/NXP edition of Cygwin, you should still install Cygwin as part of the JN5148 SDK installation as this SDK provides an updated version of the NXP Cygwin environment.

In both of the above cases, you should ensure that your path settings refer to the correct Cygwin and SDK installations, and that the paths are in the appropriate order.

Note: In addition to the above components, the JN5148 SDK Toolchain contains the device driver for the USB-to-serial cables supplied with the JN5148 evaluation kit. To use these cables, install this driver on your PC as described in Appendix B.

2.2 Toolchain Installation Procedure

To install the JN5148 SDK Toolchain on your machine:

Step 1 Remove any previous JN51xx SDK installation (other than a JN5139 SDK installation from JN-SW-4030 and JN-SW-4031, which need not be removed) from your machine using the **Uninstall** option from the Windows **Start** menu or via **Add or Remove Programs** in **Control Panel**. For more information, refer to Appendix D.

Step 2 Open Windows Explorer and check whether there is an existing **C:\Jennic** directory (or equivalent, if the SDK was previously installed somewhere other than the standard location). If there was an existing **Application** folder, or if there were extra plug-ins installed, these may still be present in the **C:\Jennic** directory. If so, delete any unwanted remnants from the **C:\Jennic** directory. Also, back up the **Application** folder if you want to re-use the application files in the new set-up.

Step 3 Start the SDK Toolchain installer from the file **JN-SW-4041-SDK-Toolchain-v*X.Y*.exe** on your machine. The Jennic Toolchain Setup wizard will start.

Step 4 Follow the on-screen instructions of the set-up wizard until you reach the **Choose Components** screen:

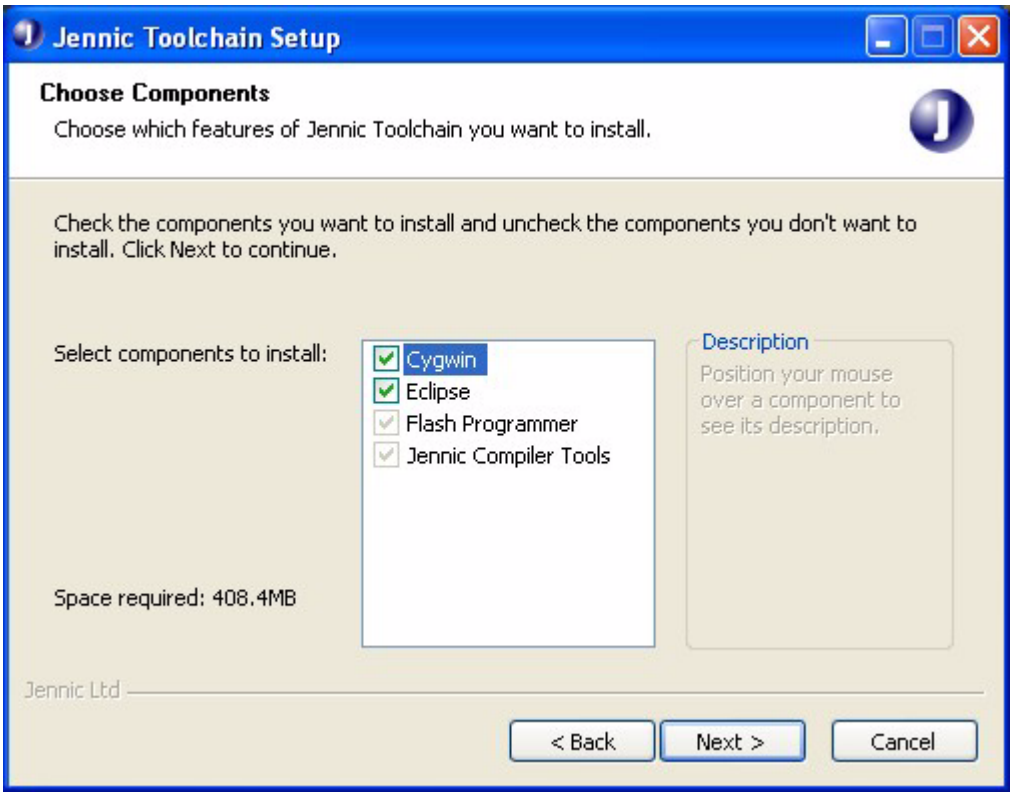

Figure 1: Choose Components Screen

By default, all the components are selected. De-select any component(s) that you do not wish to install. In particular, you should:

- de-select Cygwin if you already have a **full** Cygwin installation on your machine (see Section 2.1), otherwise leave it selected.
- de-select Eclipse if you already have the Eclipse IDE installed (although you can have more than one installation of Eclipse, if you wish).

A Cygwin installation is required on your machine, even if you wish to develop your applications using Eclipse. Refer to Section 2.1 for further information on the components.

Click **Next** to continue.

Step 5 In the next screen, choose the location where you want to install the tools:

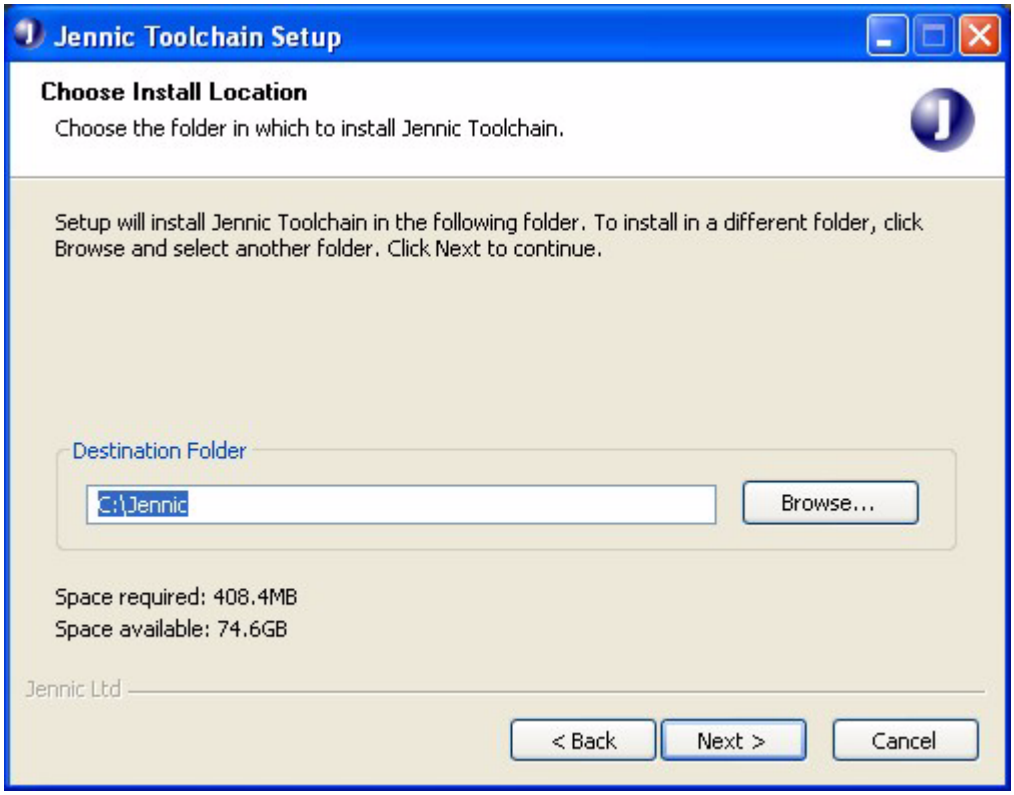

Figure 2: Toolchain Choose Install Location Screen

The set-up wizard will automatically insert the installation directory. By default, this is **C:\Jennic**. If required, you can specify another drive but must keep the **Jennic** directory (e.g. **D:\Jennic**).

Click **Next** to continue.

Step 6 In the next screen, specify the folder in which you want the installed tools to appear in the Windows **Start** menu. By default, this is set to **Jennic**.

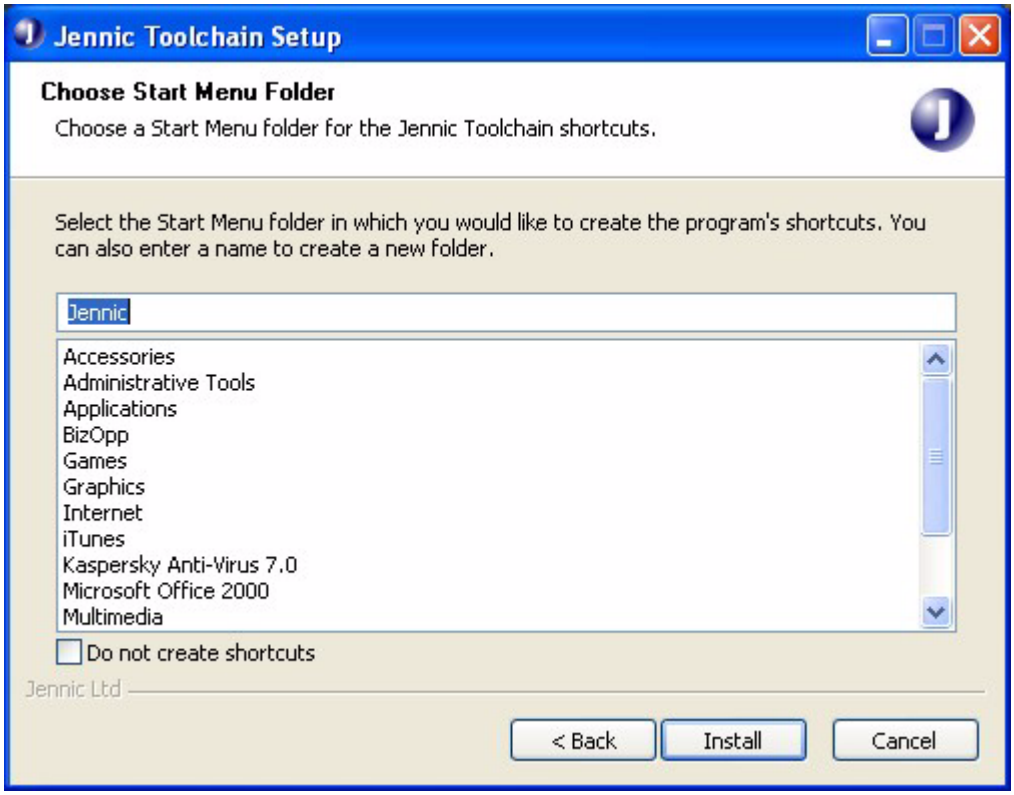

Figure 3: Choose Start Menu Folder Screen

Click **Install**.

Step 7 Wait for the installation to complete (this may take several minutes) and then click **Next** followed by **Finish**.

Step 8 Re-start your computer when prompted to do so.

Step 9 Continue to Chapter 3 in order to install the JN5148 SDK Libraries.

3. Installing the SDK Libraries

This chapter describes how to install the Libraries part of the JN5148 SDK, which must be installed after the Toolchain part (see Chapter 2). The JN5148 SDK Libraries installer (JN-SW-4040) is supplied as the following file:

JN-SW-4040-SDK-Libraries-v*X.Y*.exe

This file is available from **www.nxp.com/jennic**.

Caution: *Do not install the JN5148 SDK Libraries until you have installed the JN5148 SDK Toolchain, as described in Chapter 2.*

3.1 Contents of SDK Libraries

The software components that can be installed from the JN5148 SDK Libraries are listed in Table 3 below:

Components	Comments
ZigBee PRO APIs and stack software	Needed for applications that use ZigBee PRO
JenOS APIs	Needed for applications that use ZigBee PRO
JenNet APIs and stack software	Needed for applications that use JenNet
IEEE 802.15.4 API and stack software	Needed for all implementations
JN51xx Integrated Peripherals and Board APIs	Needed for hardware control
Configuration tool command line utilities	Needed in building applications

Table 3: Contents of SDK Libraries

3.2 Libraries Installation Procedure

To install the SDK Libraries on your machine:

Step 1 Ensure that you have installed the SDK Toolchain, as described in Chapter 2.

Step 2 Start the SDK Libraries installer from the file **JN-SW-4040-SDK-Libraries-v*X.Y*.exe** on your machine. The Jennic Libraries Setup wizard will start.

Step 3 Follow the on-screen instructions of the set-up wizard. When you reach the **Choose Components** screen, you will not be able to select individual components, since the wizard always installs all components.

Click **Next** to continue.

Step 4 In the next screen, choose the location where you want to install the libraries:

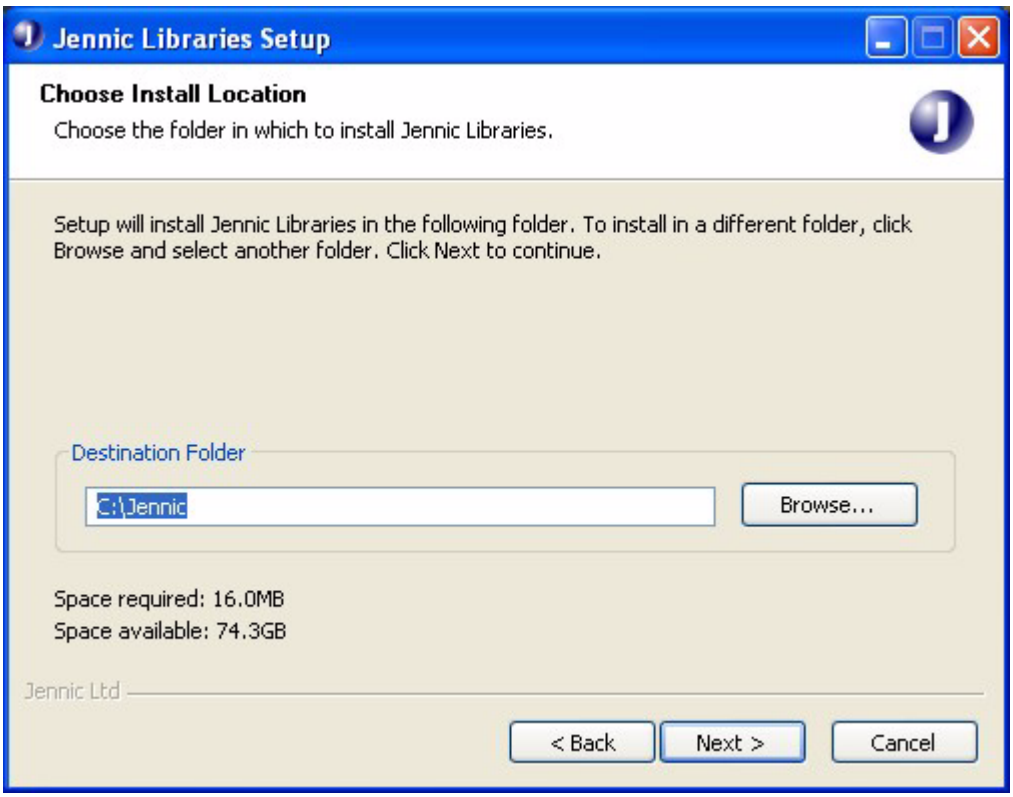

Figure 4: Libraries Choose Install Location Screen

The set-up wizard will automatically insert the installation directory. By default, this is **C:\Jennic**. If required, you can specify another drive but must keep the **Jennic** path (e.g. **D:\Jennic**).

Click **Next** to continue.

Step 5 In the next screen, specify the folder in which you want the Jennic libraries to appear in the Windows **Start** menu. By default, this set to **Jennic**.

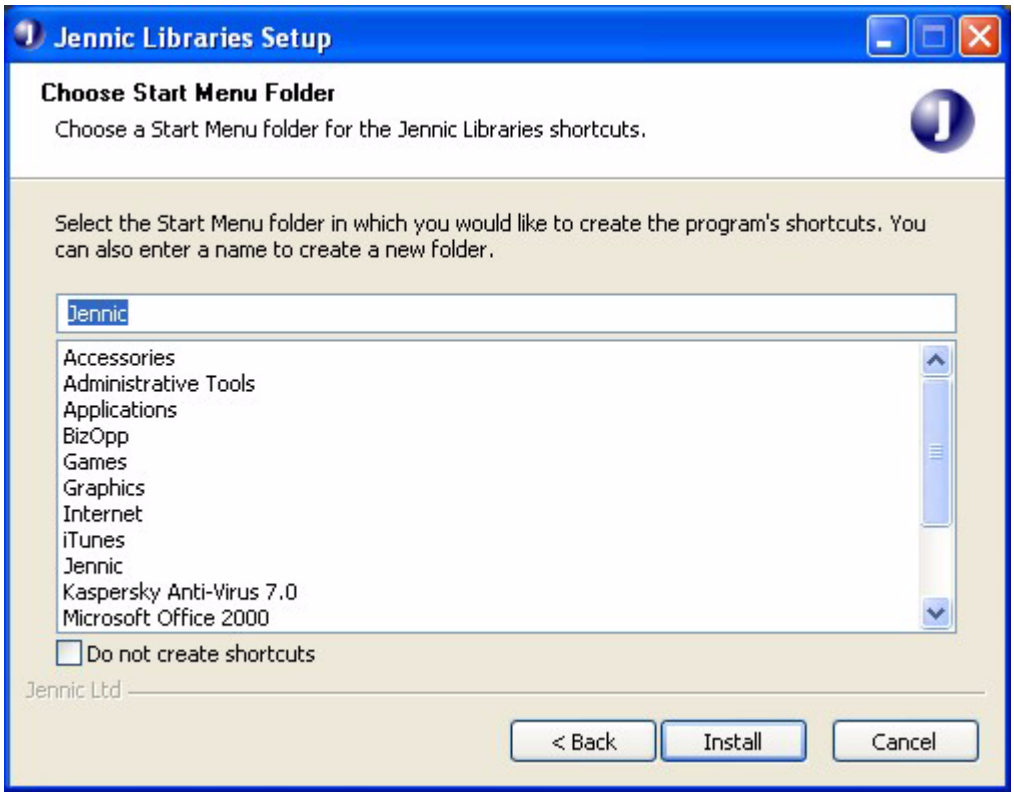

Figure 5: Choose Start Menu Folder Screen

Click **Install**.

Step 6 Wait for the installation to complete and then click **Finish**.

Step 7 Continue to Chapter 4 in order to install the required external components into Eclipse.

Part II: Eclipse Integrated Development Environment

4. Getting Started in Eclipse

The Eclipse Integrated Development Environment (IDE) is installed as a component of the JN5148 SDK Toolchain (JN-SW-4041) and is intended as the main development platform for designing wireless network applications for the NXP JN5148 microcontroller.

It is important to work through this chapter to fully prepare your Eclipse installation before embarking on your JN5148 application development within Eclipse.

4.1 Introduction to Eclipse

Eclipse is an open-source development platform, originally developed by IBM and now supported by the Eclipse Foundation (www.eclipse.org). The platform provides a fully-featured integrated environment for developing and building software applications, and is rapidly becoming the accepted standard IDE for use within the embedded software community.

The chapters of Part II: Eclipse Integrated Development Environment of this manual describe how to:

- Create an Eclipse project for your application (from an NXP template)
- Edit your application code using the Eclipse editor
- Build your application, to produce a binary file
- Download your binary file to the device that is to run the application
- Debug your application code

Note: While this manual provides guidance on using Eclipse in developing JN5148 applications, full user documentation for Eclipse is available on the Eclipse web site (www.eclipse.org).

NXP supply external tools and plug-ins for Eclipse (which are installed as described in Section 4.2). These add-ons have been developed and tested with the Ganymede version of Eclipse, which is the version supplied in the JN5148 SDK.

Caution: *NXP do not guarantee that the supplied external tools and plug-ins will work properly with any version of Eclipse other than Ganymede.*

4.2 Installing External Components into Eclipse

Once you have installed the JN5148 SDK, as described in Chapter 2 and Chapter 3, you will need to install various NXP external tools and plug-ins for the Eclipse IDE by following the procedures in this section.

The Eclipse external tools and plug-ins are outlined below.

External Tools

The external tools are provided in the SDK Toolchain and are as follows:

- Flash programmer CLI tool
- Flash programmer GUI tool
- Jennic/NXP Bash Shell
- JTAG server, to support the hardware debugger

Plug-ins (ZigBee PRO Only)

The plug-ins are configuration editors that are required for developing ZigBee PRO applications. They are not provided as part of the SDK package (how to obtain these two plug-ins is described as part of their installation procedure in Section 4.2.2). They are:

- **ZPS Configuration Editor:** This editor provides a convenient way to set ZigBee network parameters, such as the properties of the Co-ordinator, Routers and End Devices (for example, by setting elements of the device descriptors). For more information on this editor, refer to the *ZigBee PRO Stack User Guide (JN-UG-3048)*.
- **JenOS Configuration Editor:** This editor provides a graphical interface for configuring the way an application uses JenOS resources, such as timers, mutexes and ISRs. For more information on this editor, refer to the *JenOS User Guide (JN-UG-3075)*.

Note: Building an application requires the configuration tool command line utilities, which are provided in the JN5148 SDK Libraries installer (JN-SW-4040) and were installed as part of the procedure that you followed in Chapter 3.

4.2.1 Installing the External Tools

To install the external tools into Eclipse, follow the procedure below:

Step 1 Start Eclipse, either from the Windows **Start** menu or by double-clicking on the **eclipse.exe** executable file in your 'Eclipse' directory (e.g. **C:\Jennic\Tools\eclipse**).

You will be presented with a workspace selection dialogue box (see the figure below). The workspace is the directory where all your projects will be stored. It can be anywhere, but it is advised that you re-direct it to the standard development directory to keep it consistent with the JN5148 SDK, i.e. **C:\Jennic\Application**. Also, tick the box **Use this as the default and do not ask again**.

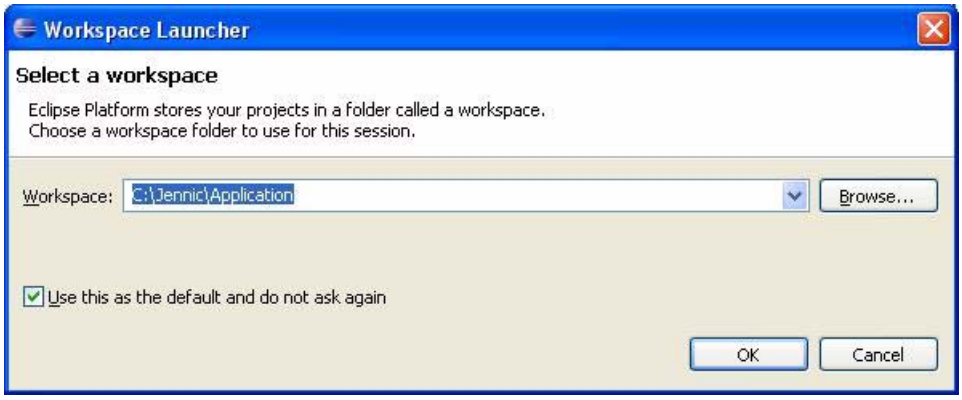

Figure 6: Workspace Launcher

Step 2 Click **OK**. If starting Eclipse for the first time, the initial start-up screen will appear.

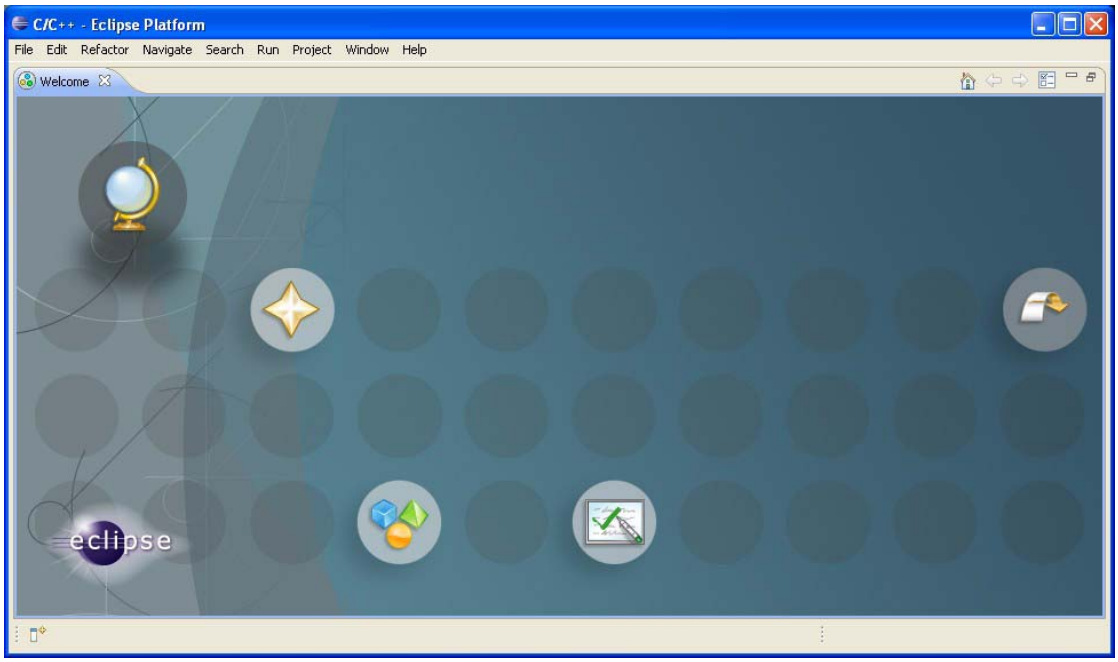

Figure 7: Eclipse Initial Start-up Screen

Step 3 Close this down by simply clicking the X on the Welcome tab. The display changes to the Eclipse main screen.

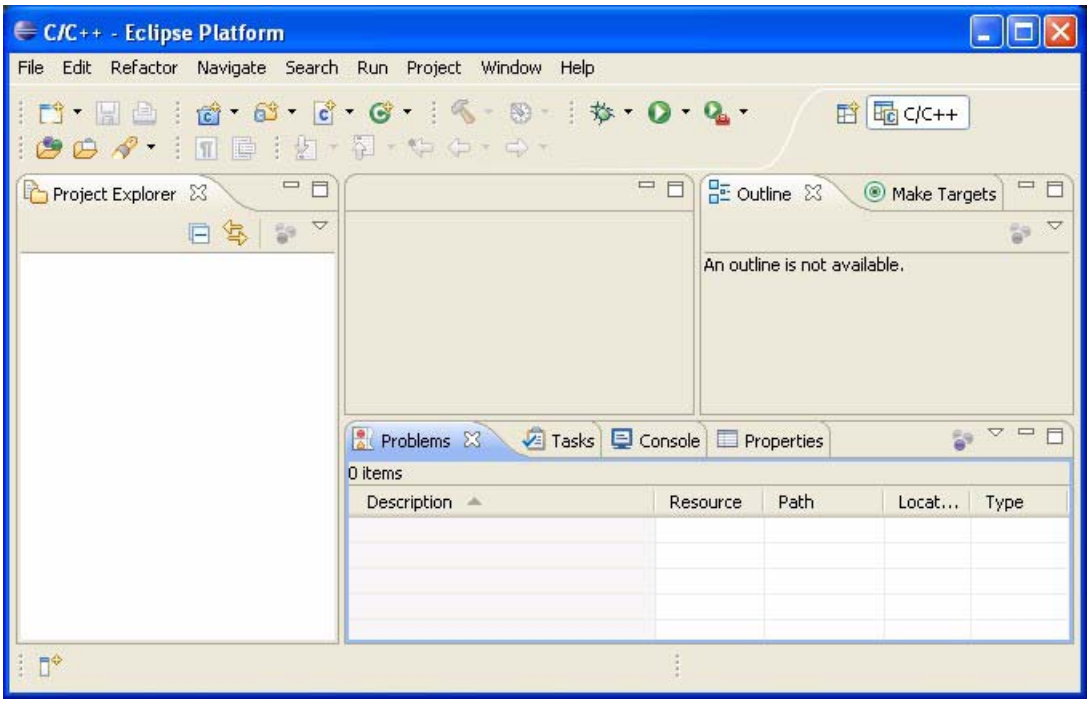

Figure 8: Eclipse Main Screen

Step 4 From the main menu, select **File > Import**. This opens the **Import** dialogue box.

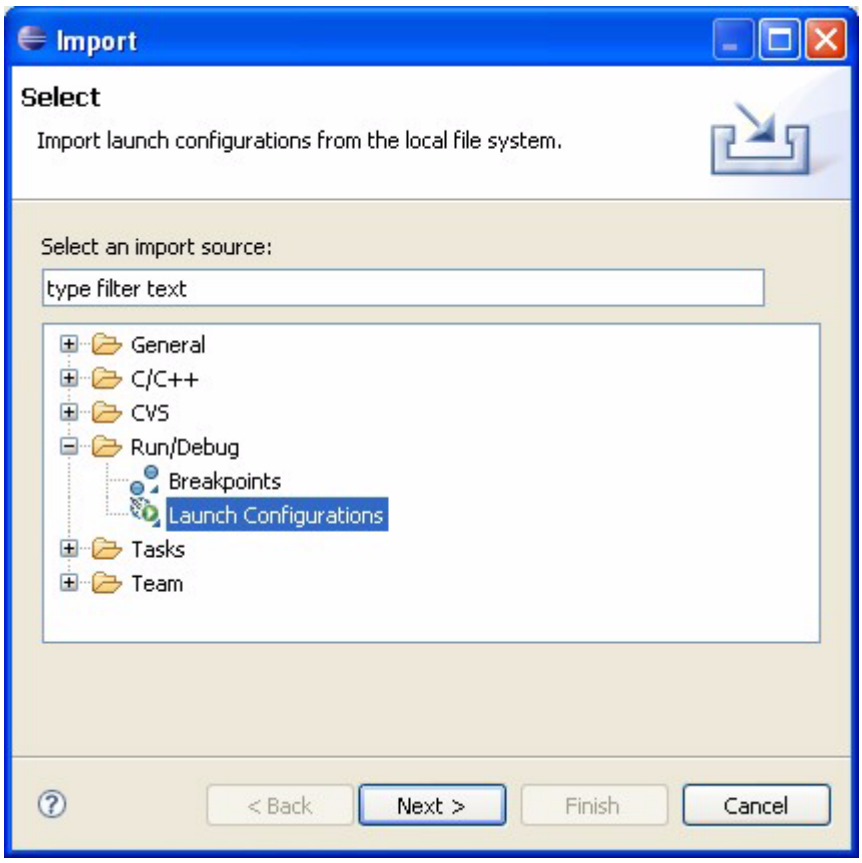

Figure 9: Import Dialogue Box

Step 5 Expand the **Run/Debug** folder and select **Launch Configurations**.

Step 6 Click **Next**. This opens the **Import Launch Configurations** dialogue box.

Step 7 Click **Browse**, browse to **C:/Jennic/Tools/eclipse_config**, click to select the folder and then click **OK**.

eclipse_config then appears in the left pane of the **Import Launch Configurations** dialogue box.

Step 8 Tick the check-box next to **eclipse_config**. All of the available **.launch** files appear in the right pane of the dialogue box - see Figure 10.

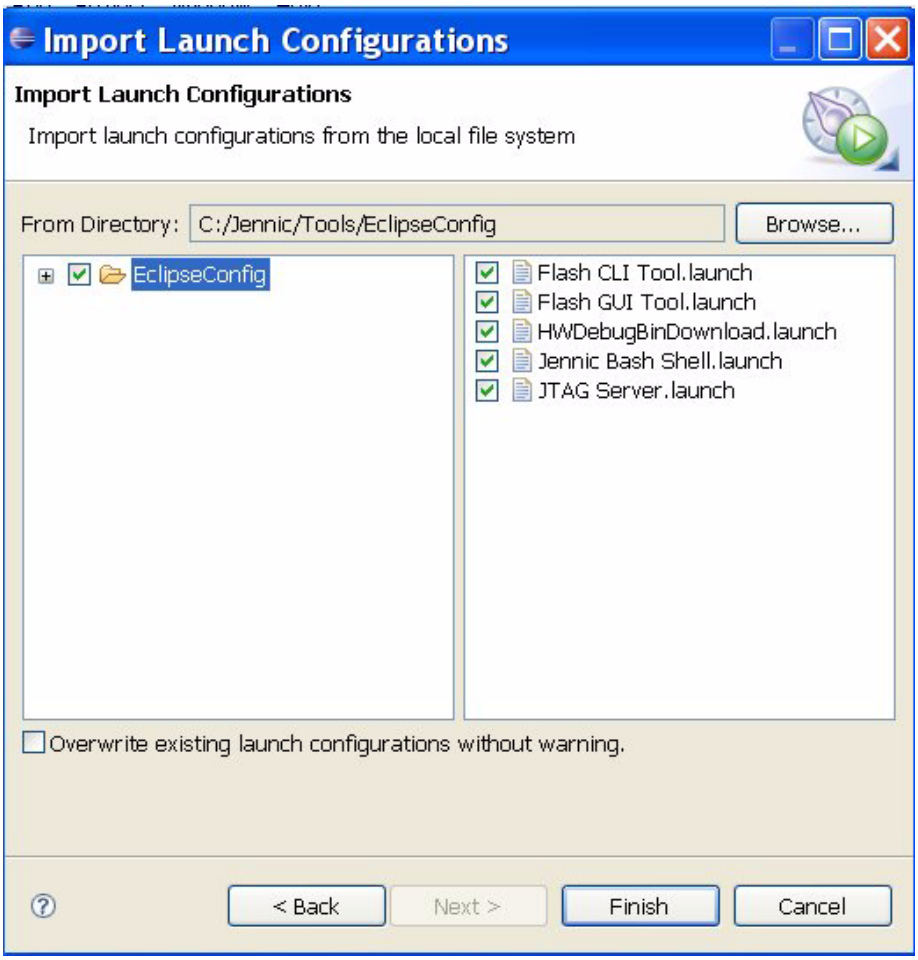

Figure 10: Import Launch Configurations

Step 9 Check that all the required **.launch** files are selected, namely:

- **Flash CLI Tool.launch**
- **Flash GUI Tool.launch**
- **HWDebugBinDownload.launch**
- **Jennic Bash Shell.launch**
- **JTAG Server.launch**

Step 10 Click **Finish**. The external tools are now automatically installed.

When the installation has finished, you should find five tools available in the **Run > External Tools** menu. They can also be accessed from the drop-down arrow next to the tools symbol  on the toolbar.

4.2.2 Installing the Configuration Editors (ZigBee PRO)

If developing ZigBee PRO applications, you will need the plug-ins for the ZPS and JenOS Configuration Editors. To install these plug-ins, follow the procedure below:

Step 1 Start Eclipse (if not already started).

Step 2 In the Eclipse main menu, select **Help > Software Updates**, then select the **Available Software** tab.

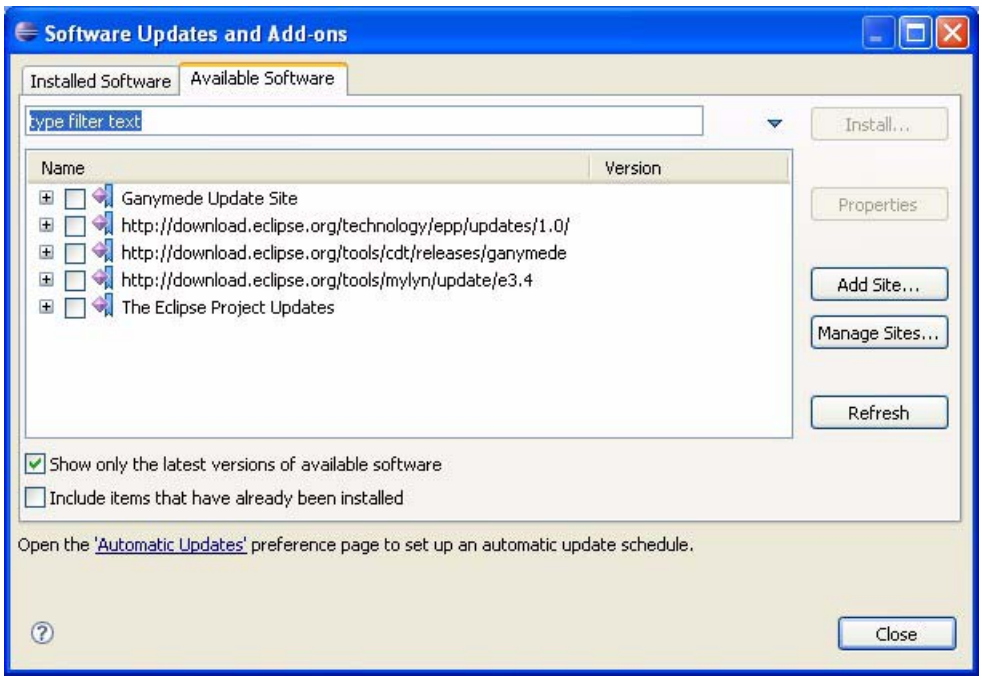

Figure 11: Available Software Tab

Step 3 Select **Add Site**.

Step 4 In the **Add Site** pop-up window, enter the location as:
http://eclipse.jennic.com/update-site

Step 5 Click **OK**. The list of plug-in sites updates automatically.

Step 6 Expand **http://eclipse.jennic.com/update-site**, then expand **Jennic ZBPro SDK**.

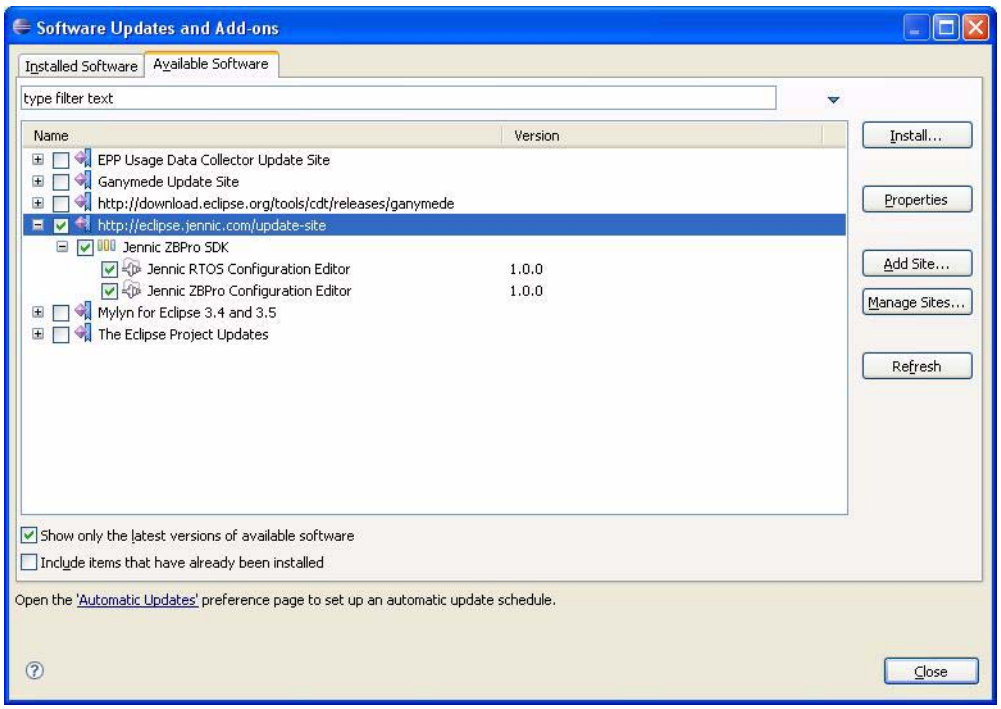

Figure 12: Jennic Software Development Kits

Step 7 Under **Jennic ZBPro SDK**, ensure that the check-boxes next to **Jennic RTOS Configuration Editor** and **Jennic ZBPro Configuration Editor** are ticked, then click **Install**.

Step 8 Wait for **Calculating requirements and dependencies** to complete - this may take a few minutes.

Note: Eclipse often begins its own update process automatically when user installations are being performed. You may see evidence of this in the bottom left-hand corner of the Eclipse screen, as it reports on the files that it has downloaded. This is the reason why Step 8 takes several minutes.

Step 9 When the **Install** window appears, click **Next**.

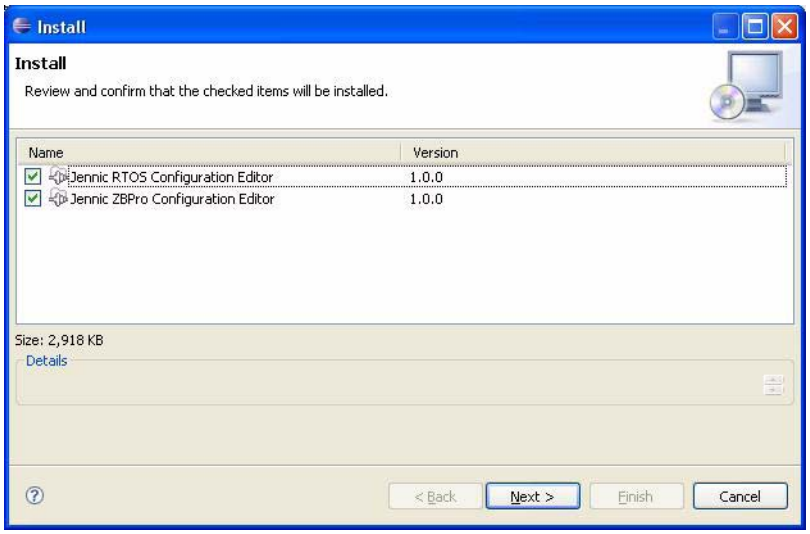

Figure 13: Software Updates Install Screen

Step 10 Read the terms of the license agreement (see the figure below) and click the button to accept.

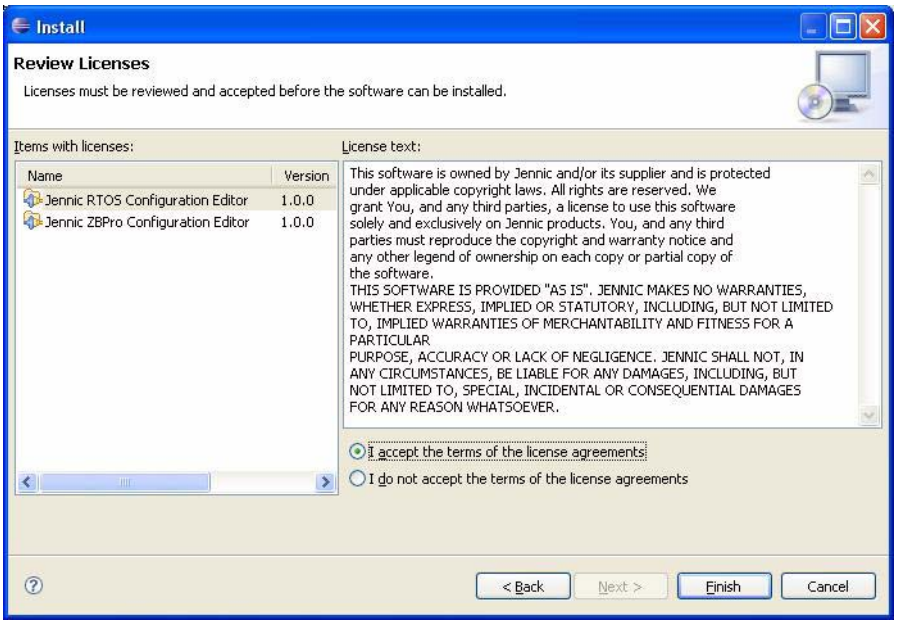

Figure 14: Review Licenses Screen

Step 11 Click **Finish** so that Eclipse installs the plug-ins that you selected. If desired, click the button for **Run in Background**.

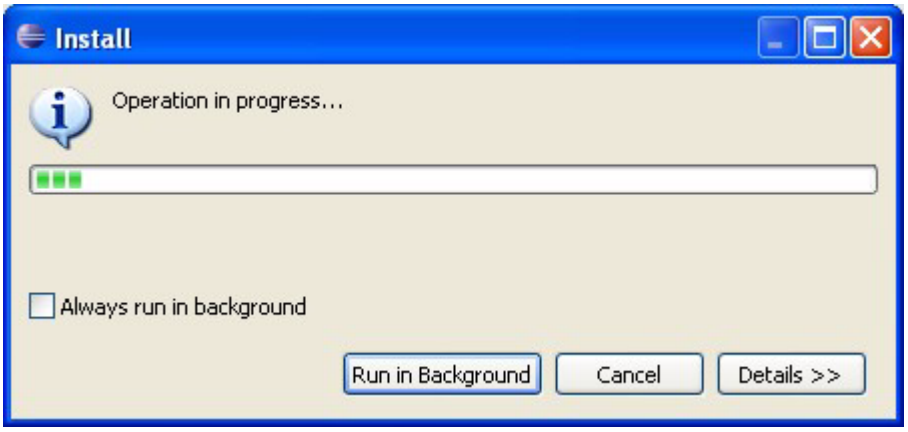

Figure 15: Installing Software Updates

Step 12 Once the plug-ins have been installed, a screen will appear which requests the acceptance of certificates for the two plug-ins. To accept them, tick the relevant boxes and click **OK**.

Step 13 Eclipse now needs to re-start to incorporate the new plug-ins. Only Eclipse itself will reboot, not the entire machine. Click **Yes** to allow the re-start.

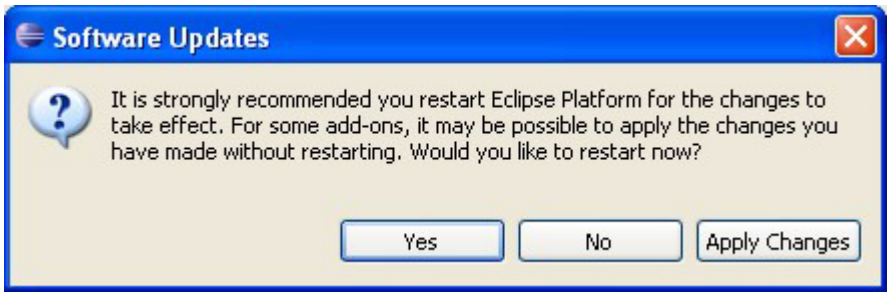

Figure 16: Software Updates Re-start Window

Step 14 You are now returned to the Eclipse main screen.

For information on how to use Eclipse, refer to Chapter 5.

For information on how to use the configuration editors, refer to the:

- *ZigBee PRO Stack User Guide (JN-UG-3048)* for the ZPS Configuration Editor
- *JenOS User Guide (JN-UG-3075)* for the JenOS Configuration Editor

5. Creating and Building Eclipse Projects

This chapter describes how to use Eclipse to create and then build your own applications to run on the JN5148 device.

It is assumed that:

- You have installed Eclipse as part of the JN5148 SDK Toolchain (JN-SW-4041), as described in Chapter 2
- You have installed the JN5148 SDK Libraries (JN-SW-4040), as described in Chapter 3
- You have installed the NXP external tools and configuration editor plug-ins for Eclipse, as described in Section 4.2

5.1 Eclipse Projects and Templates

In Eclipse, an application under development is termed a project. The creation of an Eclipse project described in this manual involves importing an NXP project template to use as a starting point. Application templates for three wireless network protocols are available from **www.nxp.com/jennic**: IEEE 802.15.4, JenNet and ZigBee PRO. The templates are supplied in the following Application Notes:

- JN-AN-1046: IEEE 802.15.4 Application Template
- JN-AN-1061: JenNet Application Template
- JN-AN-1123: ZigBee PRO Application Template

An Eclipse project folder is installed in the workspace directory that you specified when you first ran Eclipse - this should have been when you installed the NXP external tools into Eclipse, as described in Section 4.2.1.

Projects with their folders and files are displayed in a tree view in the **Project Explorer** panel on the left of the Eclipse main window. Project files can be displayed and manipulated in the Eclipse edit panel by selecting the appropriate tab.

The rest of this chapter describes:

- How to create an Eclipse project from an NXP template (see Section 5.2)
- How to work with files in an Eclipse project (see Section 5.3)
- How to build the application stored in a project (see Section 5.4)

5.2 Creating/Importing a Project (from an NXP Template)

This section describes how to create an Eclipse project for a wireless network application by importing a project template provided by NXP.

Step 1 Download the required application template (see Section 5.1) from **www.nxp.com/jennic**. Open the **.zip** file and extract it to your workspace directory, e.g. **C:\Jennic\Application**.

If using WinZip, ensure that the **Use folder names** tickbox is ticked.

Step 2 Start Eclipse, either by double-clicking on the desktop shortcut (if set up) or from the Windows **Start** menu. This should take you to your workspace that you created when you first ran Eclipse to install the NXP external tools - see Section 4.2.1.

Step 3 From the Eclipse main menu, select **File > Import**. This opens the **Import** dialogue box.

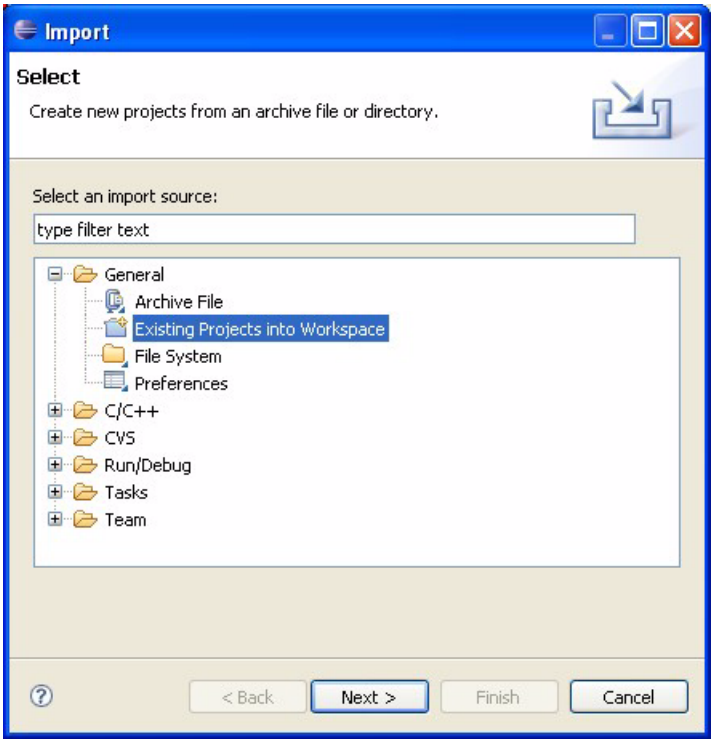

Figure 17: Import Screen

Step 4 Expand **General** and select **Existing Projects into Workspace**.

Step 5 Click **Next** and then navigate down to select your **Application** root folder in the **Browse For Folder** dialogue box.

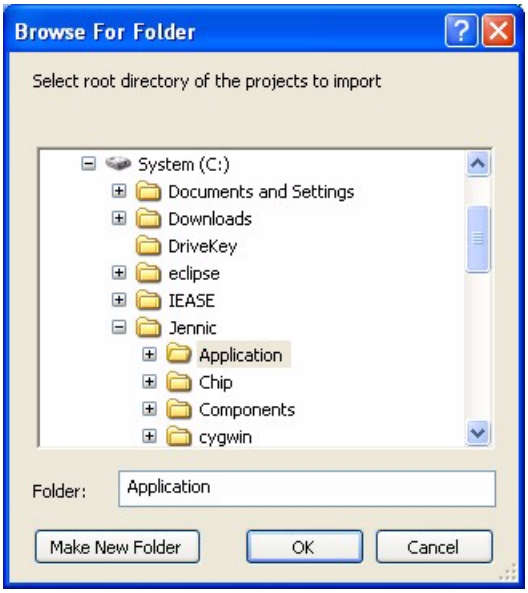

Figure 18: Browse for Application Folder

Step 6 Click **OK**.

Step 7 In the **Import** dialogue box, tick the project you want to import (untick any other projects), then confirm by clicking **Finish**. As an example, the screenshot below shows the ZigBee PRO application template.

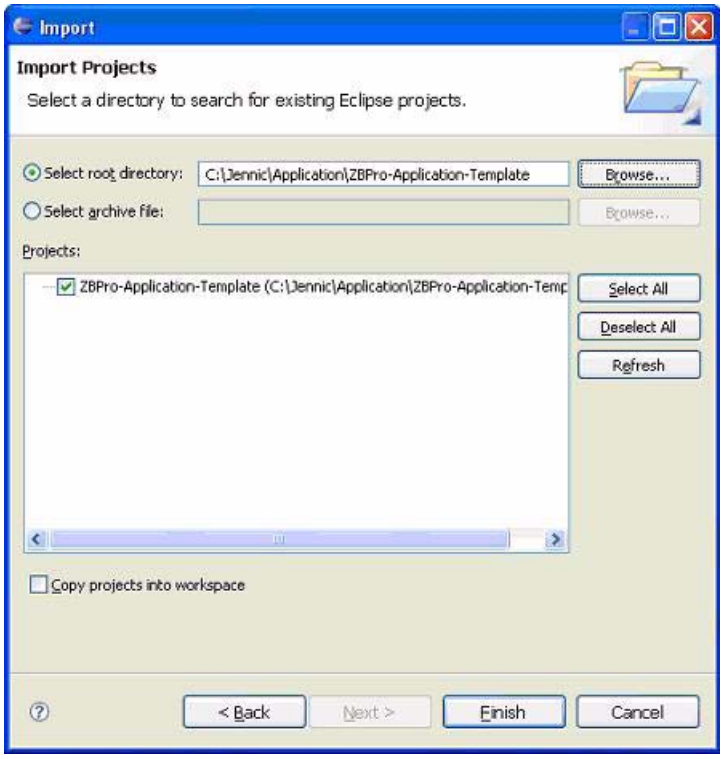

Figure 19: Import Project

Step 8 Wait a moment while the project is imported into your workspace. The project should appear in the left **Project Explorer** panel.

Your project should now include the following folders:

- **Includes** folder, containing the required library folders
- **Coordinator** folder, containing:
  - a **Build** directory which contains the makefile for the Co-ordinator
  - a **Source** directory which contains the source files for the Co-ordinator
- **Router** folder, containing:
  - a **Build** directory which contains the makefile for a Router
  - a **Source** directory which contains the source files for a Router
- **SleepingEndDevice** folder, containing:
  - a **Build** directory which contains the makefile for a Sleeping End Device
  - a **Source** directory which contains the source files for a Sleeping End Device
- **Common** folder, containing a **Source** directory which contains source files that common to the Co-ordinator, Router and End Device.
- **Doc** folder, containing the Application Note document.

Step 9 At this point, you should adapt the Eclipse project according to the needs of your application, including changing the project name:

- To re-name a project or source file, right-click on the project or file in the **Projects Explorer** view and, from the pop-up menu, select **Rename** and enter the new name. Then edit the Application Source section of the associated makefile to reflect the new name.
- To change the name of the application binary file that will result from a build, edit the associated makefile and change the Target definition as illustrated below:

  ```
  TARGET = myTargetName
  ```

- To add new source files (if any), follow the procedure in Appendix A. Then edit the associated makefile and add the new source file to the Application Source section as illustrated below:

  ```
  APPSRC += myNewSource.c
  ```


5.3 Working on Your Project

Once you have created your project, you can work on your application using Eclipse as the editor. The makefile as well as the C-code application file and any header files can be edited using Eclipse.

To edit your code, follow the procedure below:

Step 1 If the required project is not already open (if it has been closed since it was created), expand the project by clicking on the **+** symbol next to the project in the **Project Explorer** panel. This displays the project tree - see Figure 20. Similarly, click on the **+** symbol next to the **Source** folder.

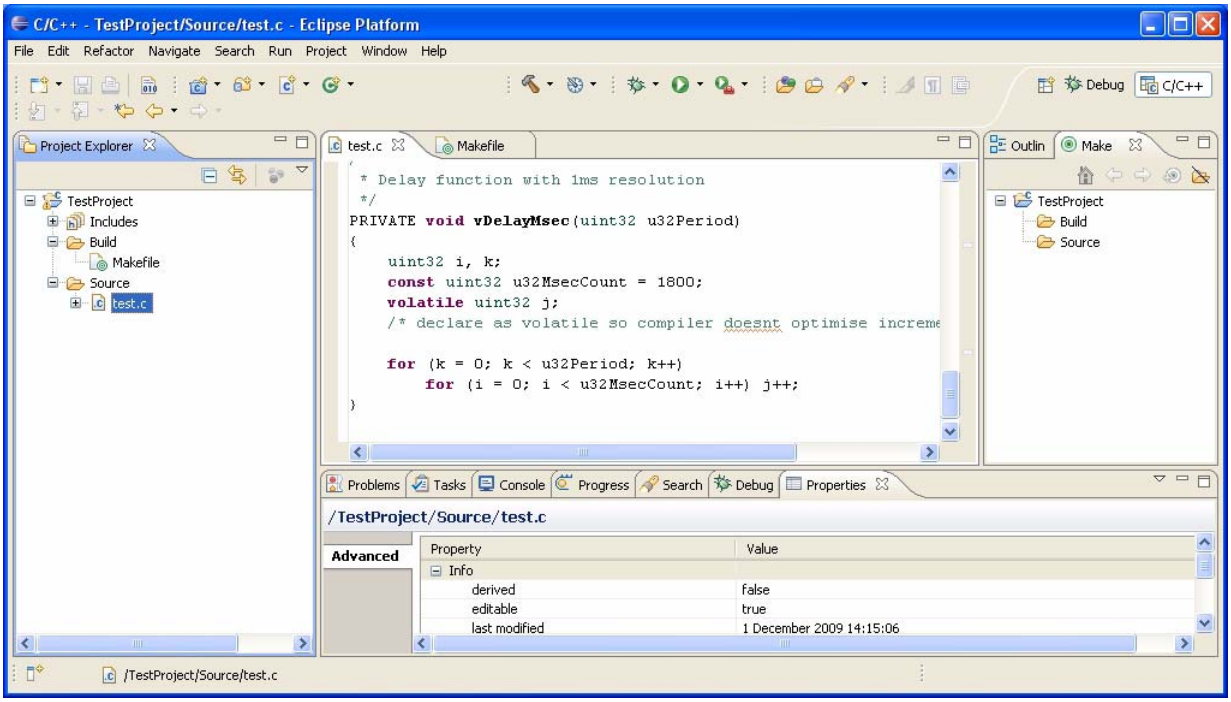

Figure 20: Editing Your C Project

Step 2 Double-click on the file to be edited in the **Project Explorer** panel. This displays a tab in the centre edit panel - see Figure 20.

Step 3 If required, rename the **.c** source file by right-clicking on it in the **Project Explorer** panel and selecting **Rename** from the pop-up menu.

The **Rename Resource** screen appears. Enter the new filename and then click **OK**.

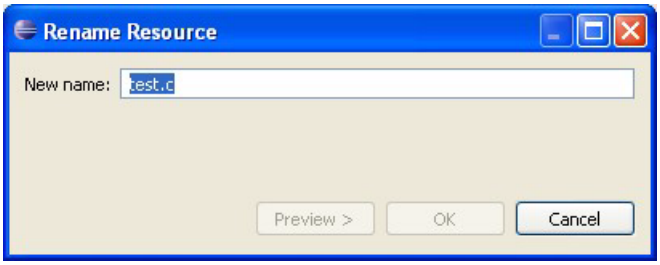

Figure 21: Rename Resource

Step 4 You can now edit the code in the main panel. If you prefer to use a different editor, right-click on the file to be edited in the **Project Explorer** panel and select **Open With** from the pop-up menu. This gives a choice of editors.

Step 5 When you have finished editing the **.c** source file, ensure that you save your changes (for example, by following the menu path **File > Save**) and then close the file (for example, by following the menu path **File > Close**).

You are recommended to save your changes regularly while editing.

Step 6 Once you have finished working on the project, save the project changes (for example, by following the menu path **File > Save All**) and close the project (for example, by following the menu path **File > Close All**).

Note: Make sure that you update the project makefile to contain the new filename specified above.

5.4 Building Your Project

Building your project can be performed simply within the Eclipse environment, as follows.

Step 1 Ensure that your makefile is present and complete (see Section 5.2), and that your editing is complete (see Section 5.3).

Step 2 Build your application by either of the following methods:

- Click the 'hammer' icon  on the toolbar - the application will then build automatically.
- In the **Projects Explorer** or **C/C++ Projects** view on the left, right-click on the relevant project and, from the pop-up menu, select **Build Project** - the application will then build automatically.

Step 3 Any errors/warnings created by the make process will be displayed in the **Problems** tab at the bottom of the screen. Standard output can be seen under the **Console** tab.

Step 4 The project binaries will be created as **.bin** files in the **Build** folder.

6. Downloading an Application Binary

Once you have built your project (as described in Section 5.4), you must download the binary output to the Flash memory attached to the JN5148 device that is to run the code. This chapter describes how to perform the download.

You must use a Flash programmer to download your application's binary file to the Flash memory of the target device. Eclipse does not have a built-in Flash programmer, but the JN51xx Flash Programmer (supplied in the JN5148 SDK Toolchain) can be run from Eclipse via the **External Tools** menu. As well as a GUI version of the Flash programmer, there is a command line version (CLI) which can be programmed for each target chip and build type (Debug or Release).

For more information about the JN51xx Flash Programmer, refer to the procedure for downloading binary code in the *JN51xx Flash Programmer User Guide (JN-UG-3007)*.

6.1 Pre-requisites

Ensure that you have the following:

- A target device containing a JN5148 microcontroller.
- A serial cable and dongle allowing connection between your PC and the target device.
- The **.bin** file to be downloaded – following a build, this file is placed in the **Build** directory for the project.

In order to access the Flash programmer from within Eclipse, the **External Tools** menu must have been set up as described in Section 4.2.1.

6.2 Download Procedure

To download your **.bin** file to a device:

Step 1 Connect a USB port of your PC to the target device using an NXP-supplied USB-to-serial cable. Make sure you connect the black wire of the cable to Pin 1 of the serial connector on the target device. If prompted to install the device driver for the USB-to-serial cable, refer to Appendix B.

Step 2 In Eclipse, follow the menu path **Run > External Tools > External Tools Configurations**.

This displays the **External Tools Configurations** dialogue box, containing a list of tools - see Figure 22.

> **Note:** You will need to set the PC communications port that has been assigned to the connection to the target device – to identify this port, refer to Appendix C.

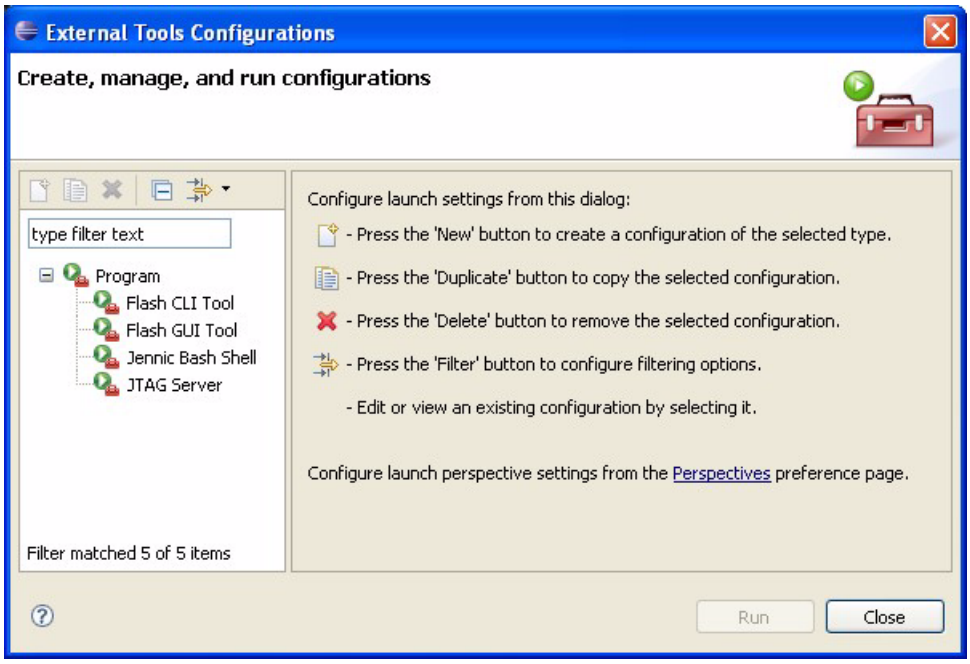

Figure 22: External Tools Configurations

Step 3 Choose the Flash programmer that you want to use - either the Flash CLI tool or the Flash GUI tool. Click to highlight it. The window changes, as illustrated in Figure 23 for the Flash GUI.

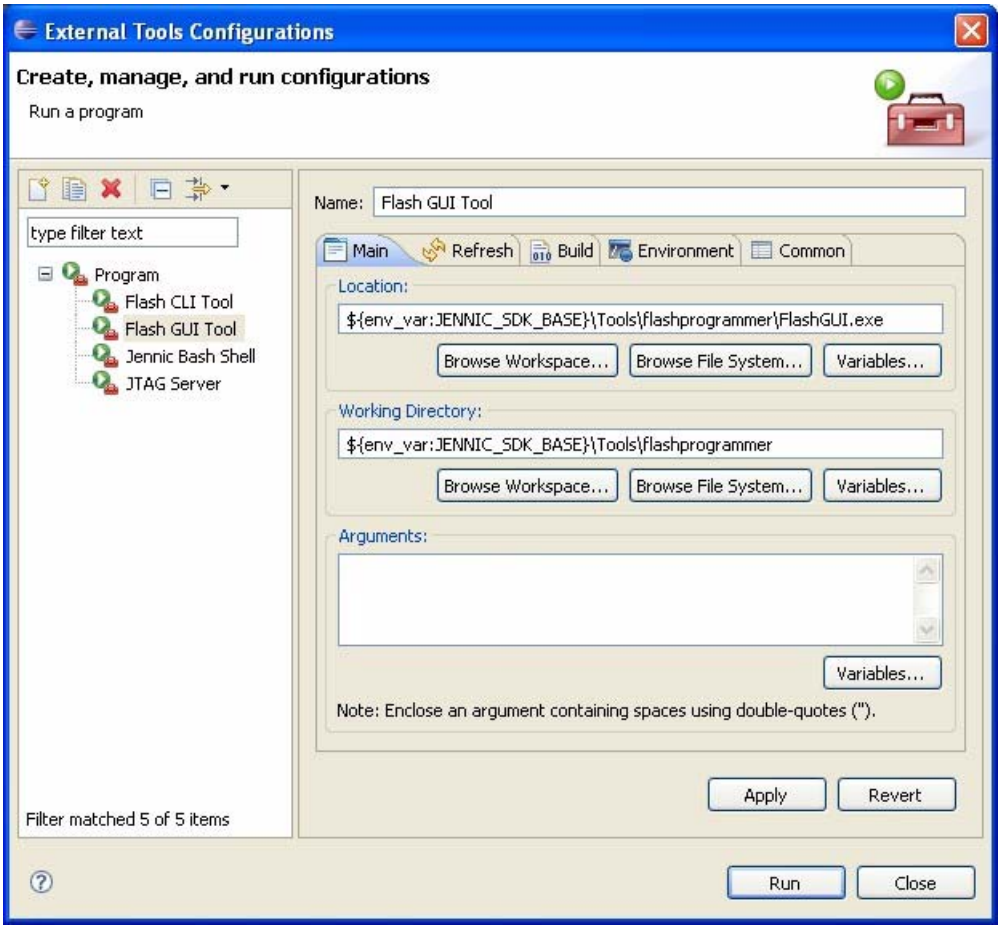

Figure 23: Flash GUI Configuration

Step 4 Click **Run** to run the tool.

> **Note:** After the tool has been run for the first time, it will appear as an option in the **Run > External Tools** menu. It is only necessary to access **Run > External Tools > External Tools Configuration** again if you need to change the parameters.

- If you selected the Flash GUI tool then the Flash Programmer GUI window is displayed. For further instructions, refer to the *JN51xx Flash Programmer User Guide (JN-UG-3007)* - you will need to continue from Step 3 of the download procedure for the Flash Programmer GUI.
- If you selected the Flash CLI tool then you will first be prompted to specify the communications port and binary file for the download - identifying the relevant port is described in Appendix C. For further instructions, refer to the *JN51xx Flash Programmer User Guide (JN-UG-3007)* - you will need to continue from Step 6 of the download procedure for the Flash Programmer CLI.

Step 5 Once the download has finished, disconnect the device from the PC and power-cycle the device.

7. Debugging Application Code

This chapter describes the available methods for debugging your application code.

In order to debug your code, you must have done the following:

1. Produced a Debug build of your project (see Section 5.4).
2. Loaded the resulting binary file into the Flash memory of the device that will run the application (see Chapter 6).

There are two possible approaches to debugging:

Using the GDB hardware debugger

To use the hardware debugger, you need to connect a JTAG hardware interface. A small (supplied) communication program is downloaded to the JN5148-based device (e.g. JN5148 evaluation kit board) and then the application binary is run directly from the PC. The JTAG hardware connects at one end to the PC via the (supplied) USB mini-connector and at the other end to the JN5148-based device. A port is selected for the JTAG interface to operate on. GDB is then started and is informed where the 'target' JTAG device is located. In this case, the hardware is controlled from a binary on the PC.

Further operational information on the GDB debugger is provided in Section 7.1.

Using the serial UART to send debug messages to a terminal

When debugging network applications in real-time, we suggest that you use the UART to send text messages to a terminal program, such as HyperTerminal, running on a PC. This involves embedding debug print-to-UART code segments in the main code of your application. This method is described in Section 7.2.

Note: The GDB hardware debugger is used via Eclipse. However, Eclipse is not required when outputting debug messages to a terminal (apart from generating and building the application to be debugged).

7.1 GDB Hardware Debugger

This section describes the principles of operation of the GDB hardware debugger and then the configuration of the debugger.

7.1.1 Principles of the GDB Hardware Debugger

This appendix explains the operation of the GDB debugger. This integrated debugger provides an easy-to-use debugging method from within Eclipse.

Using the integrated GDB debugger, you can:

- run to a breakpoint or cursor position
- watch local variables
- single-step, and step into and out of functions

Note that the GDB debugger does not allow you to:

- debug into library API code (compiled with optimisation)
- watch global variables (incompatibility between compiler and GDB)
- debug interrupts (GDB controls interrupts during a breakpoint)
- stop the underlying hardware timers and integrated peripherals

What is GDB?

GDB (GNU project debugger) is a general-purpose debugger that can be used to debug applications written in C, C++, Pascal and Fortran, amongst other languages. The debugger is available (free-of-charge) as part of the well-known GNU toolkit and is used through a text-based interface – here, this interface is provided by Eclipse.

One of the main features of the GDB debugger is its facility for remote debugging – that is, it allows you to debug code running on one platform from another platform (running GDB). For example, an application running on a microcontroller device can be debugged from a PC running GDB. In this case, the two platforms are connected via a serial port, a network link or some other method. This capability is utilised when debugging code running on a JN5148 microcontroller, allowing you to step through code, set breakpoints and view memory contents, as well as generally interact with the microcontroller.

The main disadvantage of the GDB debugger is that it is not easily adapted to embedded real-time environments, since GNU components were initially designed for developing desktop applications in Unix-type environments.

How does GDB operate?

For remote debug (as used for the JN5148 device), GDB interacts with the target application via a debug stub, which is a small, intermediate application compiled into the target system. Communication between GDB and the debug stub is implemented using the GDB Remote Serial Protocol – this is an ASCII message-based command set which supports tasks that include reading/writing from/to memory, querying

registers and running the application under test. NXP's debug stub is activated using the macro HAL_GDB_INIT.

The debugger deals with setting and processing a breakpoint in the following way:

1. When a breakpoint is set, GDB employs memory read/write commands to non-destructively replace a source instruction with a TRAP instruction.
2. When the instruction is reached during execution, control is transferred from the processor to the debug stub.
3. The debug stub notifies GDB that a breakpoint has been encountered. The user can now interact with the hardware, as required.
4. GDB sends a command corresponding to the required action to the debug stub.
5. Once the breakpoint has been dealt with, the debug stub returns control to the processor.

What happens during a breakpoint on the JN5148?

When a breakpoint is reached, the debug stub is invoked on the JN5148 target. The stub then services any requests and commands received from the GDB host. GDB stalls any application code that was running on the device processor before the breakpoint was encountered.

Note: During a breakpoint, only the JN5148 CPU is stalled. *The JN5148 integrated peripherals still run normally*, with timers running and expiring, network packets arriving and the integrated peripherals free to generate interrupts.

What happens after a breakpoint on the JN5148?

When you instruct GDB (via the Eclipse IDE) to step out of the breakpoint, the CPU interrupt handler processes all other queued interrupts generated during the period of the breakpoint. Therefore, any timer interrupts, network packets and analogue peripheral interrupts that have occurred during this period are processed. Control is then returned to the processor, resuming application execution.

Note: *Application data and peripheral hardware status may have changed during the breakpoint period*. The processing of queued interrupts may alter buffer contents and update program variables. Understanding this is key to explaining any unexpected behaviour in the application while it is being debugged.

7.1.2 Configuring the GDB Hardware Debugger

The procedure below describes how to configure the hardware debugger for use with Eclipse and how to set up the parameters to use the hardware debugger with your application.

Step 1 Connect a USB port of your PC to the target device using an NXP-supplied USB-to-serial cable. Make sure you connect the black wire of the cable to Pin 1 of the serial connector on the target device. If prompted to install the device driver for the USB-to-serial cable, refer to Appendix B.

Step 2 Connect another USB port on your PC to the JTAG box using the supplied mini-USB cable - the device driver is installed in a similar way to Step 1, refer to Appendix B.

Step 3 Start Eclipse (if not already started).

Step 4 If you are using the hardware debugger for the first time then proceed as follows, otherwise go to Step 11.

In the Eclipse main menu, select **Help > Software Updates**. Then select the **Installed Software** tab.

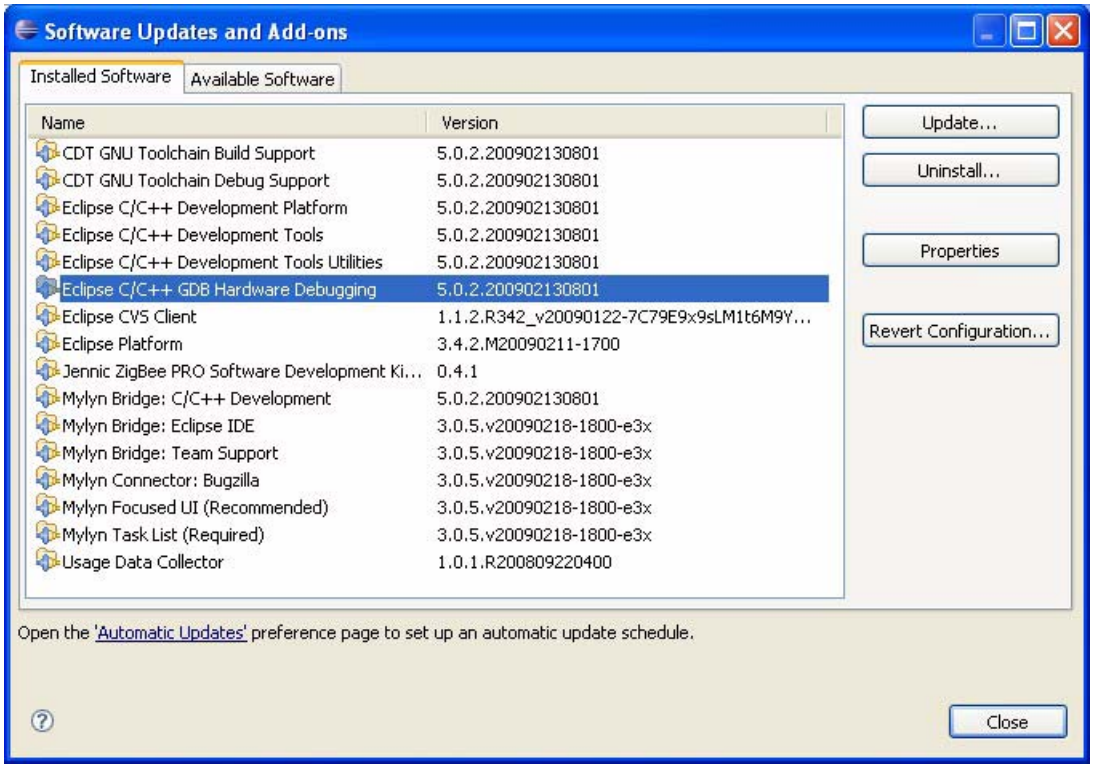

Figure 24: Checking for Hardware Debug Support

Step 5 Check that the **GDB Hardware Debugging** plug-ins are installed. If not, select the **Available Software** tab.

Step 6 Expand the option for http://download.eclipse.org/tools/cdt/releases/Ganymede. In the **CDT Optional Features** section, select **GDB Hardware Debugging** and then click **Install**.

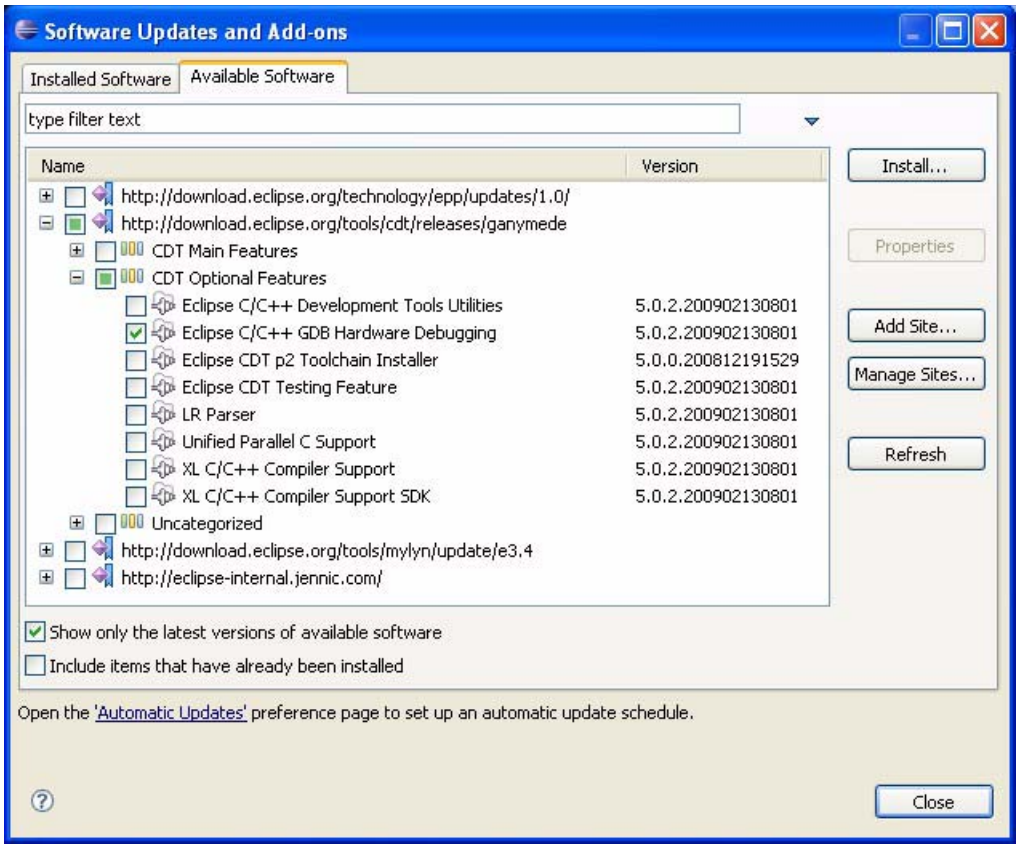

Figure 25: Software Updates Screen

Step 7 The download program resolves any dependency issues and further requirements, and then presents the **Install** screen (see Figure 26).

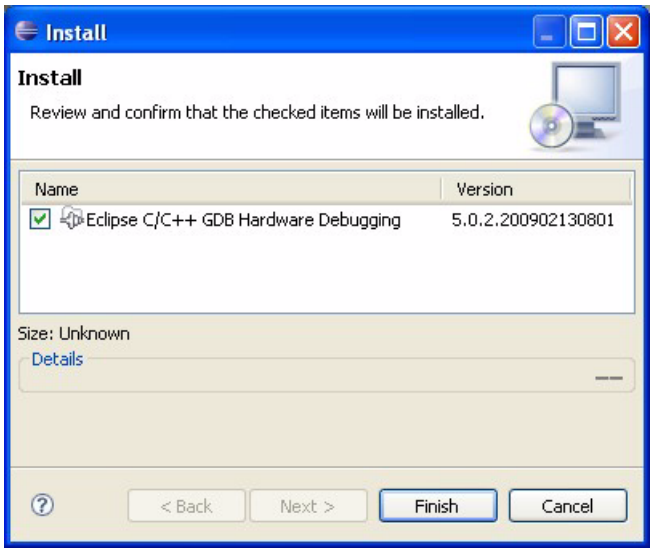

Figure 26: Software Updates Install Screen

Step 8 Click **Finish**. Eclipse then installs the plug-in that you selected.

Step 9 Eclipse now needs to restart in order to incorporate the new plug-ins. Only Eclipse itself will reboot, not the entire machine. Click **Yes** to allow the restart.

Step 10 The Eclipse main screen now re-appears.

Step 11 Click on your project in the **Project Explorer** panel to select it.

Step 12 Follow the main menu path **Run > Debug Configurations**, or click on the drop-down arrow next to the 'bug' icon and select **Debug Configurations** from the drop-down menu. This displays the **Debug Configurations** screen.

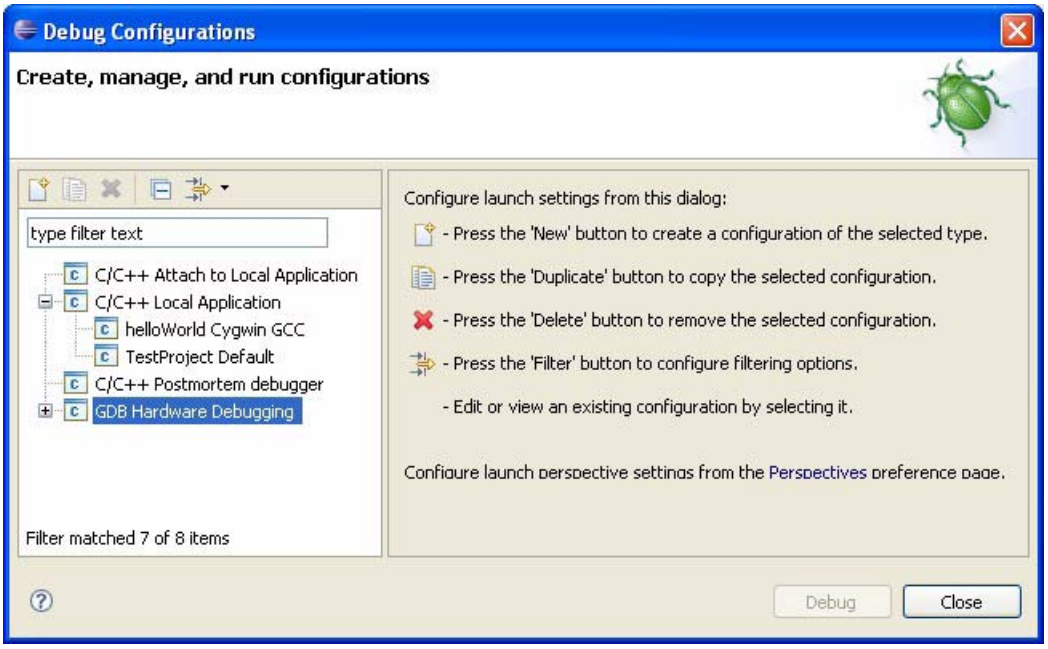

Figure 27: Debug Configurations Screen

Step 13 Highlight **C/C++ Local Application** in the left panel and press the **New** button (top left). The screen changes to a dialogue box to enter the new configuration.

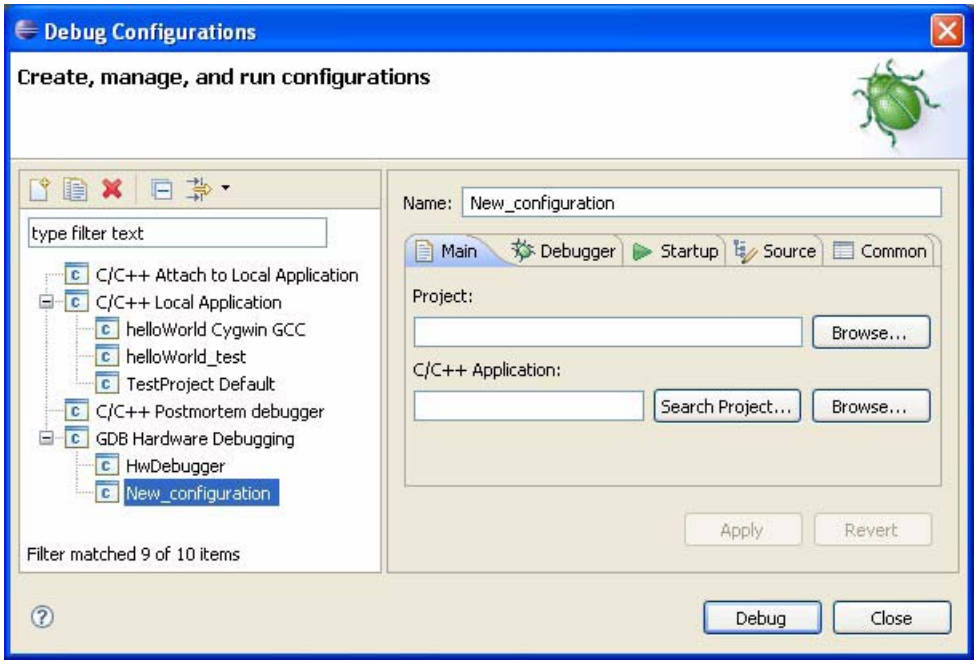

Figure 28: New Hardware Debugger Configuration

Step 14 In the **Main** tab, enter the following information (see Figure 29):

- **Name**: Enter the project configuration file name e.g. 'helloWorld_GDB_HW'. This name is for information only - it can be anything.
- **Project**: Enter the name of the specific project we are working on, e.g. 'helloWorld'.
- **C/C++ Application**: Enter a link to the project **.elf** file, e.g.

  C:\Jennic\Application\helloWorld\Build\test.elf

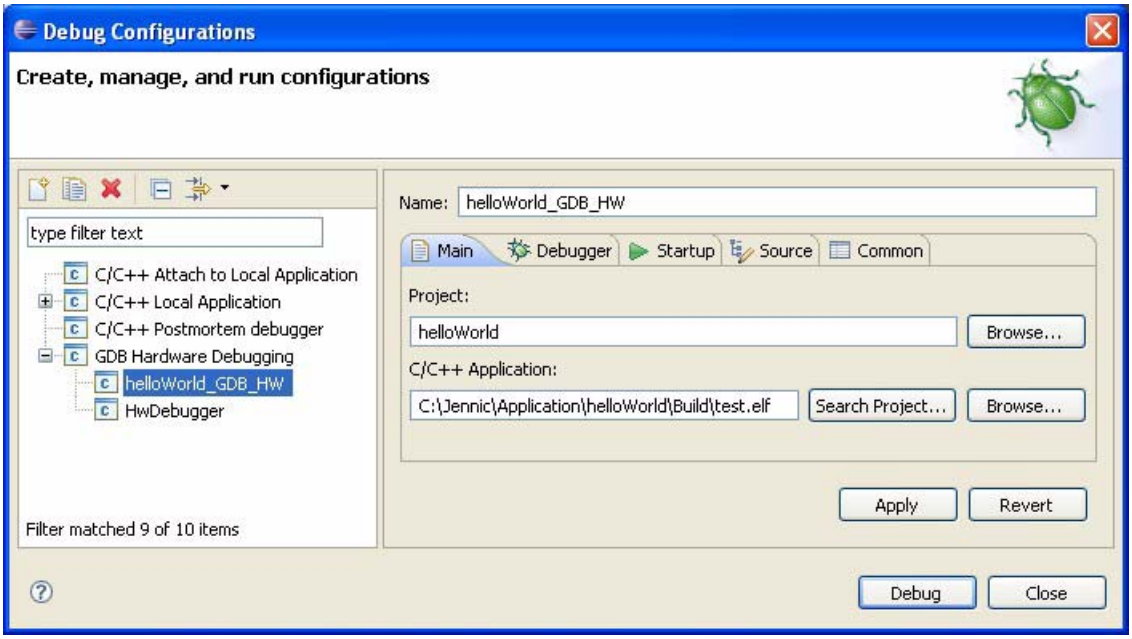

Figure 29: Hardware Debug, Main Tab

Step 15 Click **Apply** - the new configuration name appears in the left pane.
Select the **Debugger** tab.

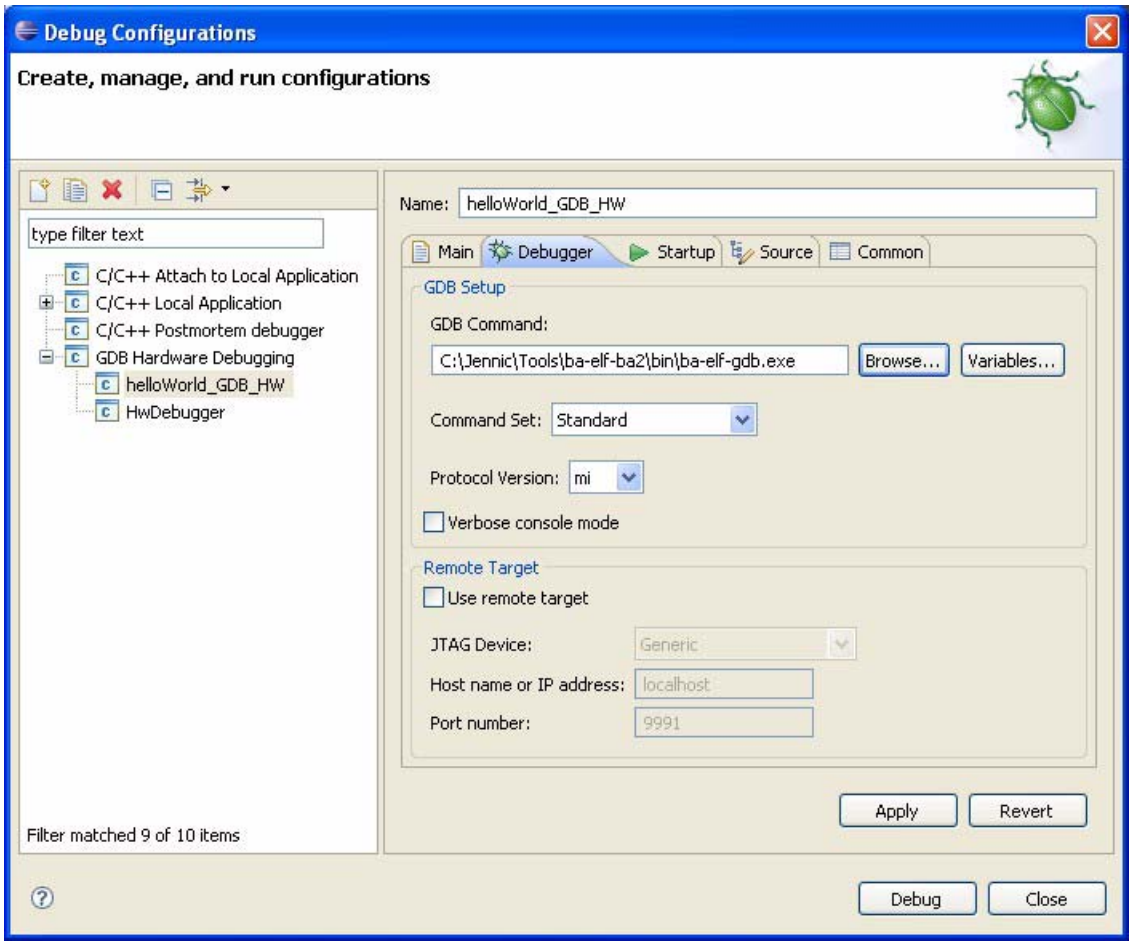

Figure 30: Hardware Debug, Debugger Tab

Step 16 In the **Debugger** tab, enter the following information:

- **GDB Command**: Browse to the ba-elf-gdb debugger located at: **C:\Jennic\Tools\ba-elf-ba2\bin\ba-elf-gdb.exe**
- **GDB command set**: Standard
- **Use remote target**: Untick the box.

Step 17 Click **Apply**, then select the **Startup** tab.

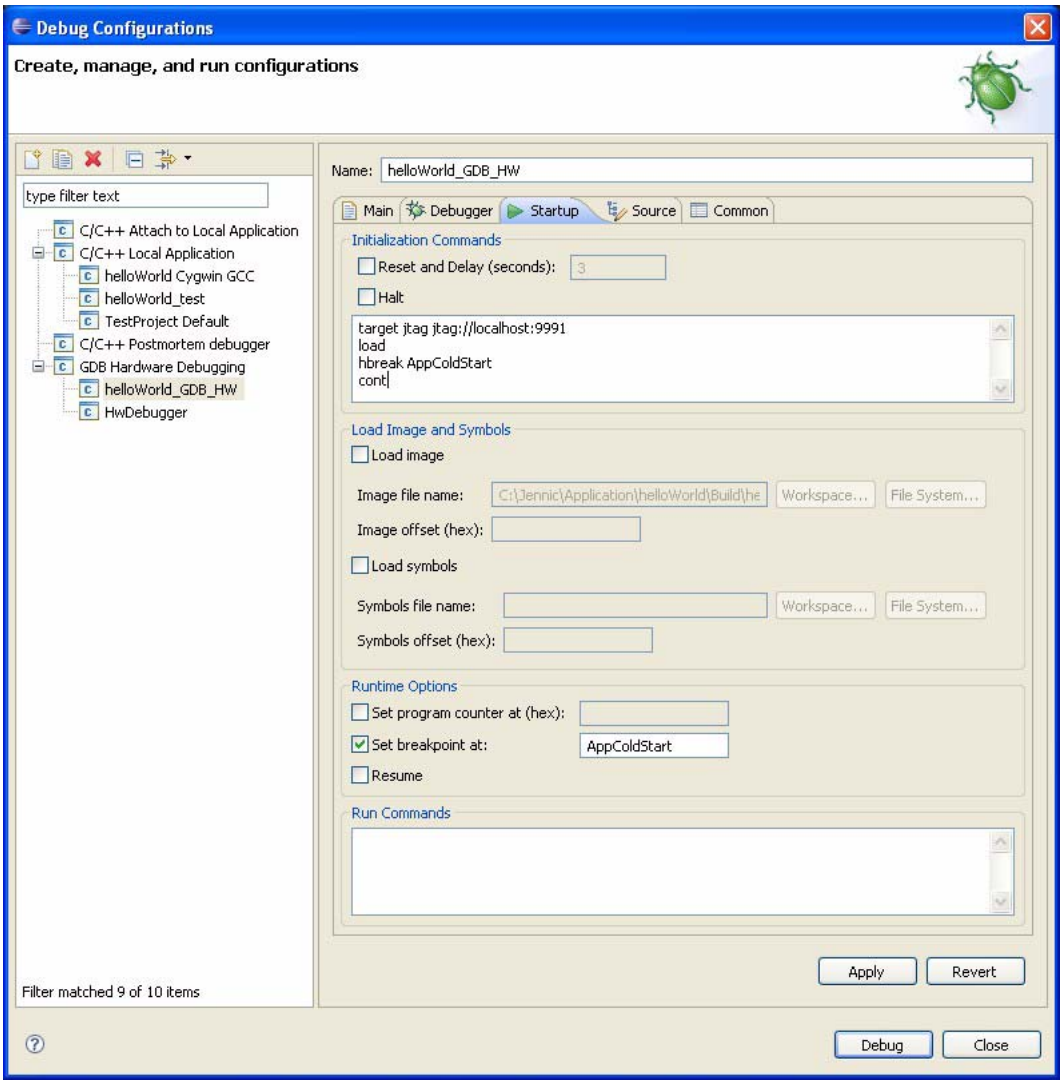

Figure 31: Hardware Debug, Startup Tab

Step 18 In the **Startup** tab, enter the following information:

- **Initialization Commands**: Untick the **Reset and Delay** and **Halt** boxes. Type the following text:
  ```
  target jtag jtag://localhost:9991
  load
  hbreak AppColdStart
  cont
  ```
- **Runtime Options**: Tick the **Set breakpoint at** box and type AppColdStart.

Step 19 Click **Apply**, then select the **Source** tab.

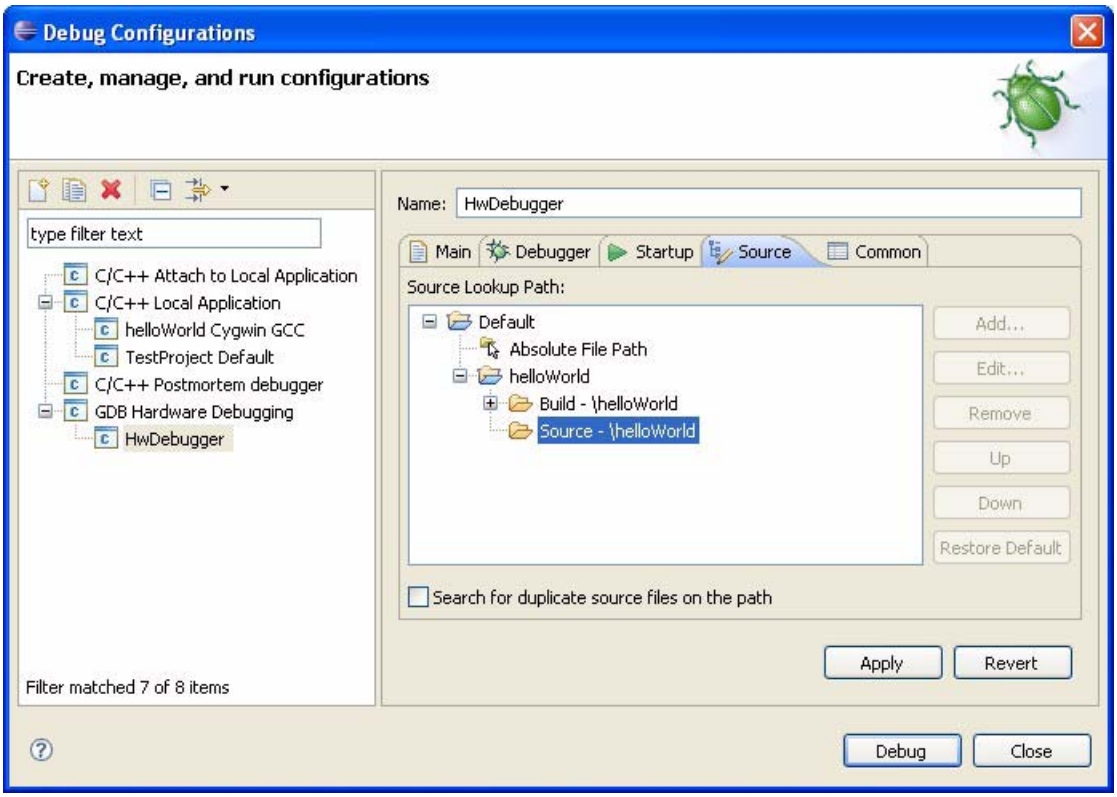

Figure 32: Hardware Debug, Source Tab

Step 20 Expand the project folder in **Source Lookup Path** and check that all the source paths relevant to your project are listed. Use the **Add** button to add additional paths, if required.

Step 21 Click **Apply**, then select the **Common** tab.

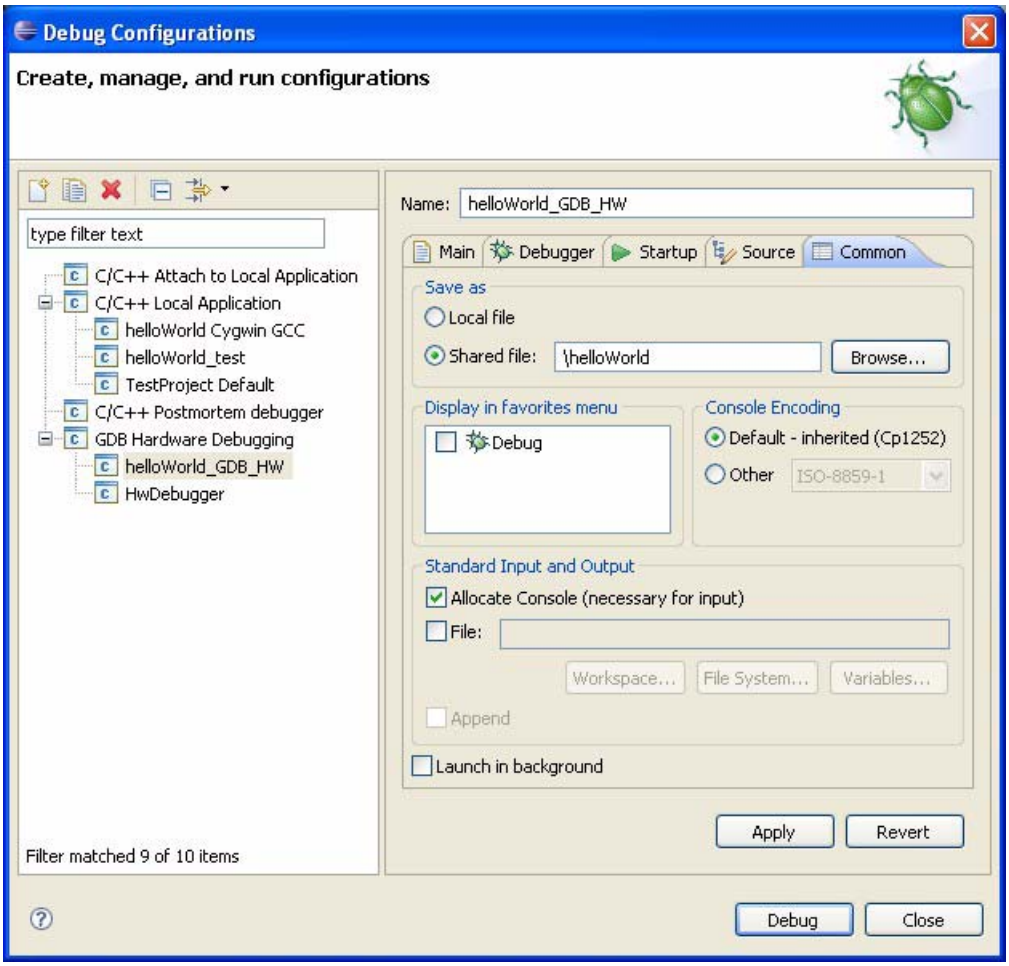

Figure 33: Hardware Debug, Common Tab

Step 22 Click the **Shared file** radio button. If the debug configuration files are to be stored in a dedicated folder then browse to that folder and select it. Otherwise leave the default, which is your project folder.

Tick the **Allocate Console** box.

Step 23 Click **Apply**, then click **Close** to register all changes and leave the debugger settings screen. The hardware debugger is now ready for use with your project - refer to Section 7.1.3.

7.1.3 Operating the GDB Hardware Debugger

This section outlines how to debug an application with the GDB hardware debugger.

The procedure below assumes that you have an application program to debug and an associated Eclipse project file.

Step 1 Start Eclipse and open the Eclipse project file or the application to be debugged.

Step 2 Ensure that the parameters for the hardware debugger are correctly set up as described in Section 7.1.2.

Verify that the **.elf** file has been built under Debug settings and resides in the **Build** directory of the project. Also verify that the corresponding **.bin** file has been downloaded to the target device using the JN51xx Flash Programmer (see Chapter 6).

Step 3 Depending on whether you are using UART0 or UART1 of the JN5148 device, use the Flash Programmer to download one of the following files to the evaluation board:

C:/Jennic/Tools/HWDebug/HWDebug_UART0_JN5148.bin
C:/Jennic/Tools/HWDebug/HWDebug_UART1_JN5148.bin

To use the Flash Programmer from within Eclipse, refer to Chapter 6.

Step 4 Once the download has completed, close the Flash programmer in order to release the serial port for the debugger to use (otherwise, GDB will not be able to access the serial port).

Step 5 Reset the device - this puts it under the control of GDB and enables Eclipse to control the device.

Step 6 From the Eclipse main menu, select **Run > External Tools > External Tools Configurations**. Click to select the **JTAG Server**.

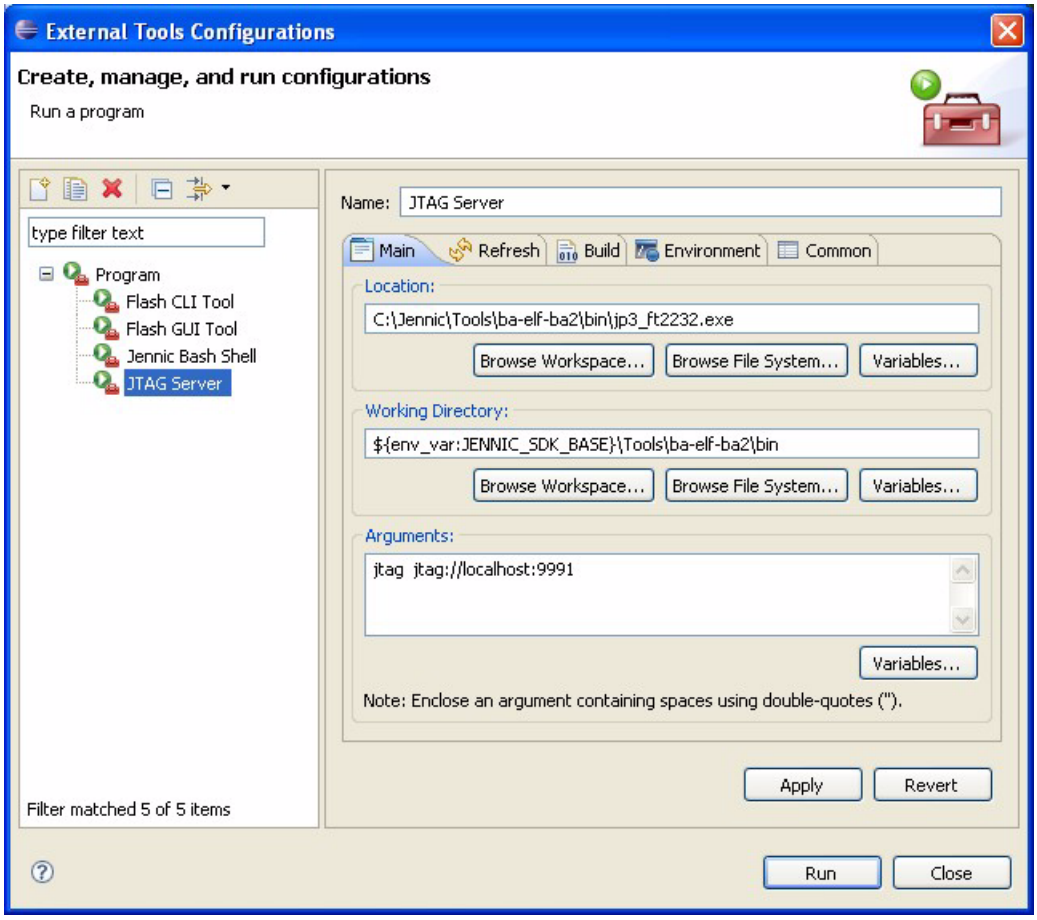

Figure 34: JTAG Server Parameters

Step 7 Check the following information in the **Main** tab:

- **Location**:
  `C:\Jennic\Tools\ba-elf-ba2\bin\jp3_ft2232.exe`
- **Working Directory**:
  `C:\Jennic\Tools\ba-elf-ba2\bin`
  or
  `${env_var:JENNIC_SDK_BASE}\Tools\ba-elf-ba2\bin`
- **Arguments**: `jtag jtag://localhost:9991`

Step 8 If the parameters are correct then click **Run** to start the JTAG server. Otherwise, correct the parameters and click **Apply** before clicking **Run**.

> **Note**: Once the JTAG server parameters are correct, the server can be started by selecting **Run > External Tools > JTAG Server** from the main menu or by selecting **JTAG Server** from the drop-down menu next to the tools icon on the taskbar.
>
> **Note**: To stop the JTAG server, select the console it is running in from the drop-down menu next to the **Display Selected Console** button and then click the red **Terminate** button .

Step 9 If you are using the debug configuration for the first time then open the **Debug Configurations** window by following the menu path **Run > Debug Configurations** or by clicking on the down arrow next to the bug symbol. Click on the required **GDB Hardware Debugging** option.

> **Note**: After your debug configuration has been run for the first time, it will appear as an option in the **Run > Debug History** menu and also as an option in the drop-down menu next to the 'bug' icon on the toolbar.

Step 10 Start the debugger for the first time by clicking on **Debug** in the **Debug Configurations** window. Subsequently, you can simply start debug by clicking on the 'bug' icon. You can also start debug by following the main menu path **Run > Debug**.

Step 11 During debugging:

- You can watch the debug progress in the **Console** tab in the lower panel.
- Use the options in the **Run** menu to toggle breakpoints and watchpoints.
- To end the debugger session, follow the menu path **Run > Debug** and stop the Debugger.

Once the debugger session has been stopped, to run the debugger again go back to Step 10 and continue from there.

7.2 Real-time Debugging via the Serial Interface

This section describes how to debug real-time network applications using the JN5148 device's serial UART to output debug information to HyperTerminal running on a PC. In this case, Eclipse is used to generate the code that is run, but does not have a role in the debug process.

`vPrintf`, which is a small memory footprint version of `printf`, is used to send formatted debugging text from the application to the UART. This reduced version of `printf` is limited to the following commands:

- %d - show a decimal value
- %x - show a value in hex
- %b - show a value in binary
- %c - show a character
- %s - show a string
- %% - show a % character

The `Printf` source and include files can be found in the directory **C:\Jennic\Components\Utilities**.

In the example presented in the sub-sections below, the debug serial text is sent via UART0, which is also used for the Flash programmer.

7.2.1 Preparing the Application

You must first prepare your application source code, as described in the procedure below.

Step 1 Load the project to be debugged into Eclipse.

Step 2 Check that the build target is present in the **Build** folder.

Step 3 To add the **Printf.c** library files, first click to select your project in the **Project Explorer** pane. From the main menu, select **File > New > File from Template**. This opens the **New File** screen.

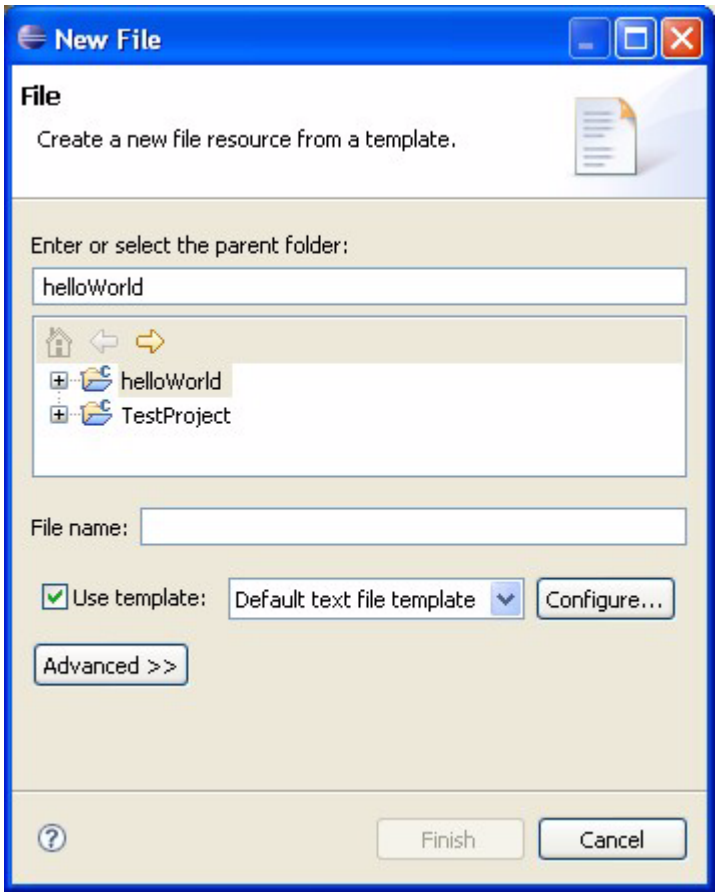

Figure 35: Adding a New File

Step 4 Expand your project by clicking on the **+** symbol then click on the **Source** folder to highlight it.

Step 5 Click on **Advanced**.

Step 6 Click on the **Link to file in file system box** to select it, then click on **Browse** and in the **Select Link Target** window select:

C:\Jennic\Components\Utilities\Source\Printf.c

The screen should appear as follows:

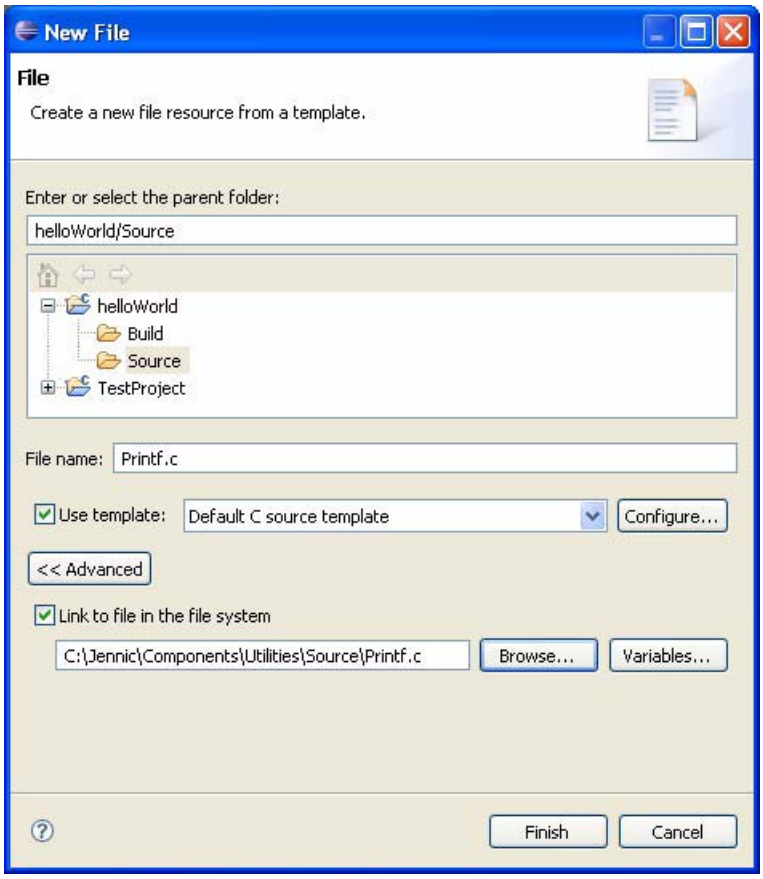

Figure 36: Adding the Printf.c File

Step 7 Click **Finish**.

Step 8 Similarly, repeat Step 3 to Step 7 to add the **Printf.h** library files. In Step 6, select **C:\Jennic\Components\Utilities\Include\Printf.h**.

You should now see both the **Printf.c** and **Printf.h** libraries in your **Source** folder.

Step 9 Add the following code to the source file under test.

a) Include the **Printf.h** header file:

```
#include "C:\Jennic\Components\Utilities\Include\Printf.h"
```

b) Initialise the UART during the hardware initialisation

```
/* Initialise serial comms unless debug mode*/
#ifndef GDB
vUART_printInit();
#endif
```

c) Add debug lines wherever they are required; for example:

```
vPrintf("\n\r\n\rAddress = %x", u16NodeId);
```


Note: `\n\r` is used to provide CR LF in the terminal emulator.

Step 10 Build the project as described in Section 5.4.

Step 11 Connect the Flash programmer, via the serial cable, to UART0 on the target device.

Step 12 Using the Flash programmer, download the application binary to the target device.

Step 13 Close the Flash programmer (otherwise it will hold the serial port open).

Note: You must connect/disconnect after each session in order to use the Flash programmer. Alternatively, you can use Bray's free Terminal v1.9 - this utility detects if another program is using the port and will automatically disconnect if you tick the **Auto Dis/Connect** box (www.hw-server.com/software/termv19b.html).

7.2.2 Configuring HyperTerminal

This section describes how to configure HyperTerminal for real-time debugging via a serial interface.

Note: HyperTerminal is not available in Windows Vista. An alternative is to use TeraTerm, which is a free download from http://www.ayera.com/teraterm/.

Step 1 Open HyperTerminal by the following the Windows **Start** menu path **All programs > Accessories > Communications > HyperTerminal**.

Note: If no modem has been configured on the PC, you may get screens requesting the location. Ignore these screens.

Step 2 Access the **New Connection** screen by following the menu path **File > New Connection**.

Type a name for the connection, then click **OK**.

Figure 37: Connection Description Screen

The next screen, **Connect To**, is then displayed.

Step 3 In the **Connect To** screen, choose the serial communications port that the board is connected to and then click **OK**.

Figure 38: Connect To Screen

The next screen, **COM Properties**, is then displayed.

Step 4 In the **COM Properties** screen, set the port properties to 19200 bits per second, 8 data bits, no parity, 1 stop bit and no flow control, then click **OK**.

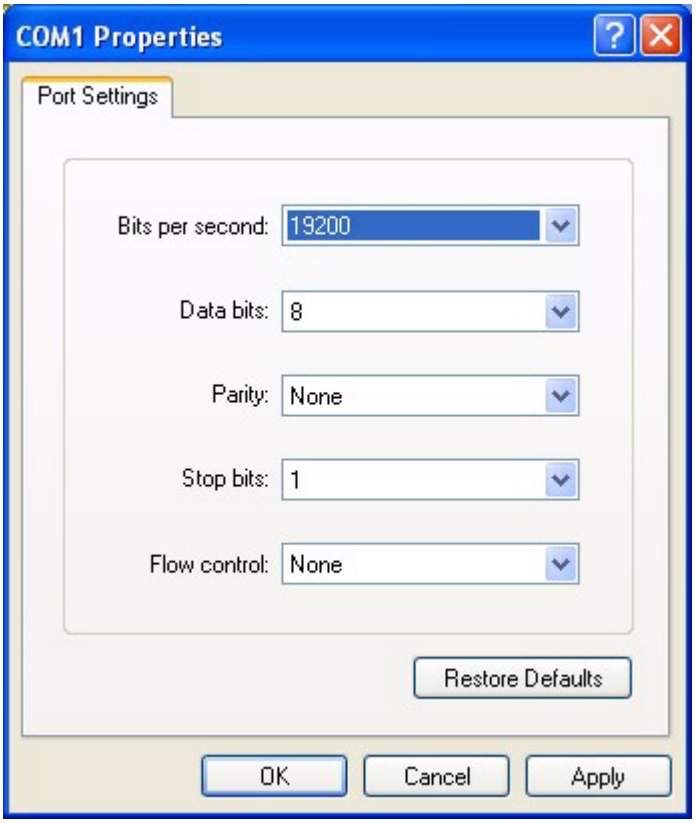

Figure 39: COM Properties Screen

The terminal connects to the communications port and the **HyperTerminal** screen is displayed.

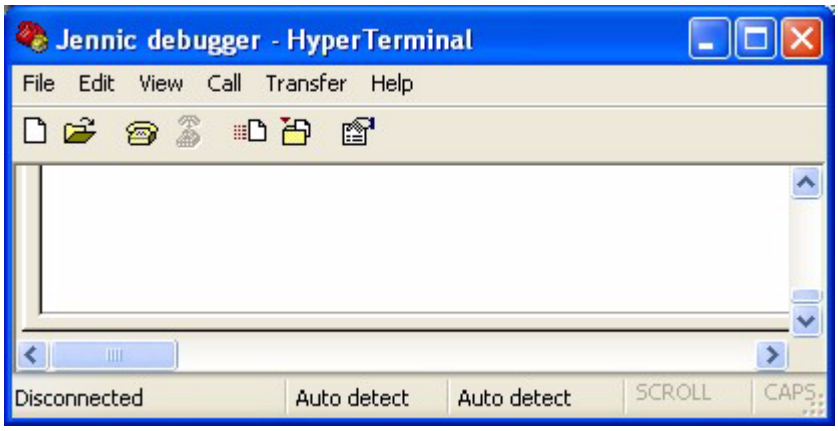

Figure 40: HyperTerminal Screen

7.2.3 Using the Serial Debugger

Run the target software by resetting the host device. The debugging trace will then appear on the HyperTerminal screen.

The terminal could also be used to send commands to the target in order to test its operation. This would require the addition of an interrupt handler to process the received commands.

An example of handling serial data is provided in the Application Note *Serial Cable Replacement using 802.15.4 (JN-AN-1005)*.

Part III: Appendices

A. Creating an Eclipse Project Source File

The procedure below describes how to add a new C source file to an Eclipse project.

Step 1 In your project in Eclipse, expand the project name folder so that the required **Source** folder (in which the new source file will go) is visible and click on it to highlight it.

Step 2 To add a file to the **Source** folder, from the main menu select **File > New > Source File**. The **New Source File** dialogue box appears. As an example, the screenshot below shows a new source file called **test.c**.

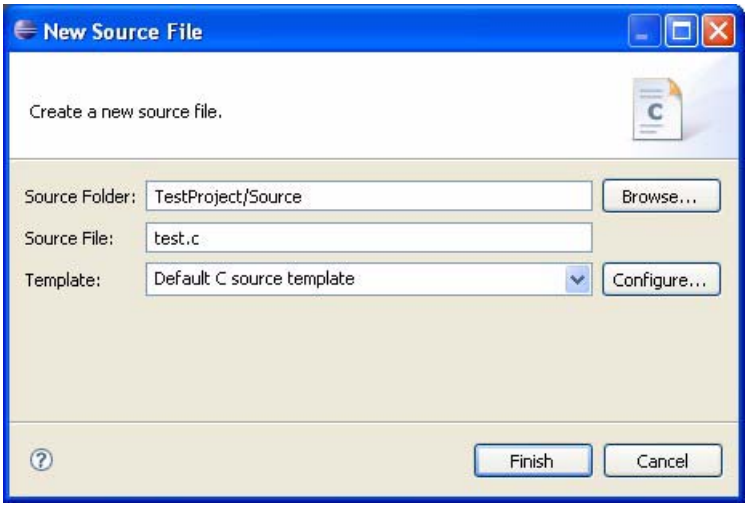

Figure 41: New Source File

Step 3 Enter the parameters as follows:

- **Source Folder**: This field should be automatically completed.
- **Source File**: Enter the name of the source file you want to create, e.g. **test.c**.
- **Template**: Select **Default C source template** from the drop-down menu.

Step 4 Click **Finish**. The new source file appears in the **Project Explorer** panel.

Step 5 The content of your new source file can be viewed and edited by clicking on the tab (e.g. **test.c**) in the centre panel.

Step 6 Edit the source file, as required.

B. Installing the USB-to-Serial Cable Driver

The USB-to-serial cable supplied with the JN5148 evaluation kit allows a PC USB port to be used as a serial communications port and requires an FTDI driver. This driver is provided in the JN5148 SDK Toolchain (JN-SW-4041) and must be installed on your PC the first time you use the supplied cable – for example, when downloading binary code from a PC to a board. This installation is described below (although you may not need this procedure if Windows automatically finds the required driver on the Internet).

Step 1 When you plug the USB-to-serial cable into a USB port of your PC, check whether **Found new hardware wizard for TTL232r-3v3** is displayed.

If this appears, you must install the driver by following the rest of this procedure. Otherwise, the driver is already installed.

Step 2 Fill in the screen **Install from a specific location**, as follows:

a) Select the radio button **Search for the best driver in these locations**.

b) Tick the checkbox **Include this location in the search**.

c) Using the **Browse** button, navigate to the directory **FTDI_drivers** in the installed SDK on your PC:

C:\Jennic\Tools\Drivers\FTDI_drivers

d) Click **OK**.

The wizard will automatically fill in the details in the drop-down search box.

Step 3 In the **Found new hardware wizard** screen, click **Next**.

Step 4 Wait for the wizard as it searches for and installs the new driver. On completion, it will display the message “Completing the Found new hardware wizard”. Click **Finish** to complete.

In some cases, you may need to repeat the procedure from Step 2, depending on your hardware configuration.

Finally, the **Found new hardware** bubble will indicate that the hardware is installed and ready for use.

Note: Alternatively, you can obtain the relevant driver for your operating system from the FTDI web page **www.ftdichip.com/FTDrivers.htm**. Go to the VCP drivers, download the required driver to your desktop and double-click on its icon to install.

C. Identifying the PC Communications Port Used

When connecting your PC to a board, you need to find out which serial communications port your PC has allocated to the connection, as described below.

Step 1 In the Windows Start menu, follow the menu path:

Start > Control Panel > System

This displays the **System Properties** screen.

Step 2 In the **System Properties** screen:

a) Select the **Hardware** tab.

b) Click the **Device Manager** button

This displays the **Device Manager** screen.

Step 3 In the **Device Manager** screen:

a) Look for the **Ports** folder in the list of devices and unfold it.

Identify the port which is connected to the board (it will be labelled 'USB Serial Port') and make a note of it (e.g. COM1).

D. Uninstalling the SDK

This appendix describes how to uninstall a JN51xx SDK that has been installed using an SDK Libraries installer (JN-SW-4030 or JN-SW-4040) and an SDK Toolchain installer (JN-SW-4031 or JN-SW-4041).

Note: This method does not apply to earlier SDK versions that use installers JN-SW-4026 and JN-SW-4027. To remove these SDK versions, you should use **Add or Remove Programs** in the Windows **Control Panel**.

You can remove the SDK from your machine using the uninstallers provided in the Jennic folder in the Windows **Start** menu.

***Caution**: If you are uninstalling the SDK in order to install the latest version, you should first back up your **SDK/Application** folder before installing the new SDK, otherwise you will lose your existing applications.*

Step 1 Follow the Windows **Start** menu select path **Start > All programs > Jennic** or **Start > Programs > Jennic > JN-SW-404x products** and then select:

- **Uninstall JN-SW-40XX-SDK-Libraries** or **Uninstall Jennic Libraries** to remove the SDK Libraries
- **Uninstall JN-SW-40XX-SDK-Toolchain** or **Uninstall Jennic Toolchain** to remove the SDK Toolchain

Step 2 Open Windows Explorer and check if the **C:\Jennic** folder has been completely removed. If not, any remaining elements can be deleted manually.

In the Windows **Start** menu, check if all unwanted items in the **Start > Programs > Jennic** path have been removed. If not, the unwanted elements can be deleted manually.

Revision History

Version	Date	Comments
1.0	7-July-2009	First release
1.1	12-Mar-2010	Title of manual and SDK changed Jenie/JenNet added to protocol options Hardware debug .bin download added to Eclipse launch configurations Updates made concerning Cygwin installation advice and other minor issues
1.2	14-May-2010	References to Jenie/JenNet patch removed
1.3	17-Jun-2010	Minor modifications/corrections made
2.0	22-Nov-2010	Incorporated information from former *Eclipse IDE User Guide (JN-UG-3063)*

Important Notice

NXP Laboratories UK Ltd
(Formerly Jennic Ltd)
Furnival Street
Sheffield
S1 4QT
United Kingdom

Tel: +44 (0)114 281 2655
Fax: +44 (0)114 281 2951
E-mail: info@jennic.com

For the contact details of your local Jennic office or distributor, refer to the Jennic web site:

www.nxp.com/jennic