

The background of the top half of the page features a futuristic, blue-toned digital landscape. It includes glowing lines, a grid of light points, and a bright light source on the left creating a lens flare effect. A large, semi-transparent blue sphere is visible on the right side.

# Jennic

TECHNOLOGY FOR A CHANGING WORLD

## **Eclipse IDE User Guide**

JN-UG-3063  
Revision 1.3  
20 May 2010



---

## Contents

<b>About this Manual</b>	<b>5</b>
Organisation	5
Conventions	5
Acronyms and Abbreviations	6
Related Documents	6
Feedback Address	6
<b>1. Overview</b>	<b>7</b>
<b>2. Creating and Building Your Own Projects</b>	<b>9</b>
2.1 Eclipse Projects and Templates	9
2.2 Creating/Importing a Project (from a Jennic Template)	10
2.3 Working on Your Project	14
2.4 Building Your Project	15
<b>3. Downloading an Application Binary</b>	<b>17</b>
3.1 Pre-requisites	17
3.2 Download Procedure	18
<b>4. Debugging Application Code</b>	<b>21</b>
4.1 GDB Hardware Debugger	22
4.1.1 Configuring the Hardware Debugger	22
4.1.2 Operating the GDB Hardware Debugger	32
4.2 Real-time Debugging via the Serial Interface	35
4.2.1 Preparing the Application	35
4.2.2 Configuring HyperTerminal	38
4.2.3 Using the Serial Debugger	41
<b>Appendices</b>	<b>43</b>
<b>A. The Integrated Debugger Explained</b>	<b>43</b>
<b>B. Creating a Project Source File</b>	<b>46</b>
<b>C. Installing the USB-to-Serial Cable Driver</b>	<b>47</b>
<b>D. Identifying the PC Communications Port Used</b>	<b>48</b>



---

## About this Manual

This User Guide is designed to help you use the Eclipse IDE (Integrated Development Environment) to develop applications for devices in the Jennic wireless microcontroller family. The guide focuses on tasks required to implement applications on Jennic devices and does not cover all the features and functionality of Eclipse.

---

## Organisation

This manual consists of four chapters and three appendices, as follows:

- [Chapter 1](#) provides a brief overview of using the Eclipse IDE.
- [Chapter 2](#) describes how to create, develop and build your own Eclipse projects (using templates provided by Jennic).
- [Chapter 3](#) describes how to download your executable code to the Flash memory of the Jennic wireless microcontroller.
- [Chapter 4](#) explains how to debug your application.
- The [Appendices](#) cover topics including the integrated debugger and tasks related to connecting a PC to a Jennic wireless network device.

---

## 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` 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
JenOS	Jennic Operating System
SDK	Software Development Kit
ZPS	ZigBee PRO Stack

---

## Related Documents

- [1] JN5148 Software Developer's Kit Installation Guide (JN-UG-3064)
- [2] JN51xx Flash Programmer User Guide (JN-UG-3007)

---

## Feedback Address

If you wish to comment on this manual, or any other Jennic user documentation, 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  
Jennic Ltd  
Furnival Street  
Sheffield S1 4QT  
United Kingdom  
[doc@jennic.com](mailto:doc@jennic.com)

---

## 1. Overview

For those engineers who wish to develop software using an Integrated Development Environment (IDE), Jennic recommends the Eclipse IDE. Eclipse is a fully-featured, open-source development environment available free-of-charge and is rapidly becoming the accepted standard IDE for use within the embedded software community.

Eclipse is available as part of the Jennic JN5148 Software Developer's Kit (SDK) Toolchain (JN-SW-4041). In order to develop wireless network applications for the JN5148 device, you will also need the JN5148 SDK Libraries (JN-SW-4040). Installation instructions for the SDK are provided in the *JN5148 Software Developer's Kit Installation Guide (JN-UG-3064)*. The SDK installers and the Installation Guide are available from the support area of the Jennic web site ([www.jennic.com/support](http://www.jennic.com/support)).

This manual describes how to:

- Create an Eclipse project for your application
- 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





---

## 2. Creating and Building Your Own Projects

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

It is assumed that:

- You have installed Eclipse as part of the JN5148 SDK Toolchain (JN-SW-4041)
- You have installed the Jennic external tools and configuration editor plug-ins for Eclipse
- You have installed the JN5148 SDK Libraries (JN-SW-4040)

Installation instructions for all of the above are provided in the *JN5148 Software Developer's Kit Installation Guide (JN-UG-3064)*.

---

### 2.1 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 a Jennic project template to use as a starting point. Application templates for three wireless network protocols are available from the Jennic web site: IEEE 802.15.4, Jenie/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: Jenie 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 Jennic external tools into Eclipse, as described in the *JN5148 Software Developer's Kit Installation Guide (JN-UG-3064)*.

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 a Jennic template (see [Section 2.2](#))
- How to work with files in an Eclipse project (see [Section 2.3](#))
- How to build the application stored in a project (see [Section 2.4](#))

## 2.2 Creating/Importing a Project (from a Jennic Template)

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

- Step 1** Download the required application template (see [Section 2.1](#)) from the Support area of the Jennic web site ([www.jennic.com/support](http://www.jennic.com/support)). 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 Jennic external tools - see the *JN5148 Software Developer's Kit Installation Guide (JN-UG-3064)*.

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

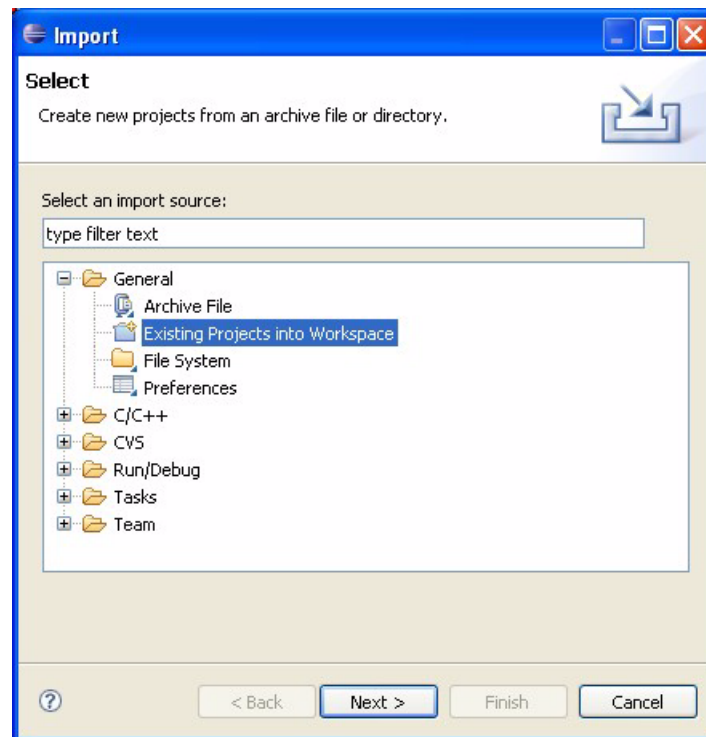
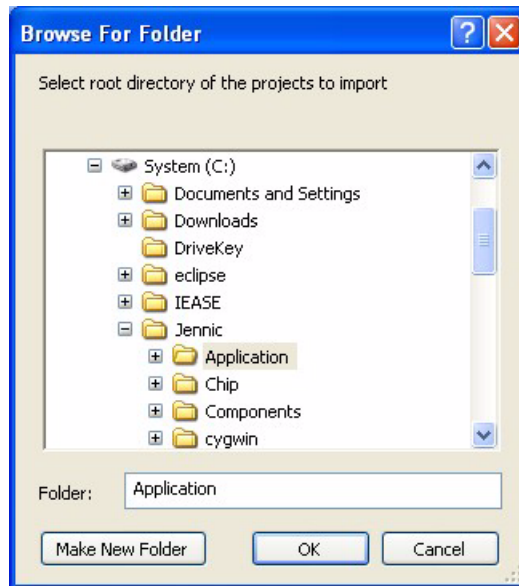


Figure 1: Import Screen

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

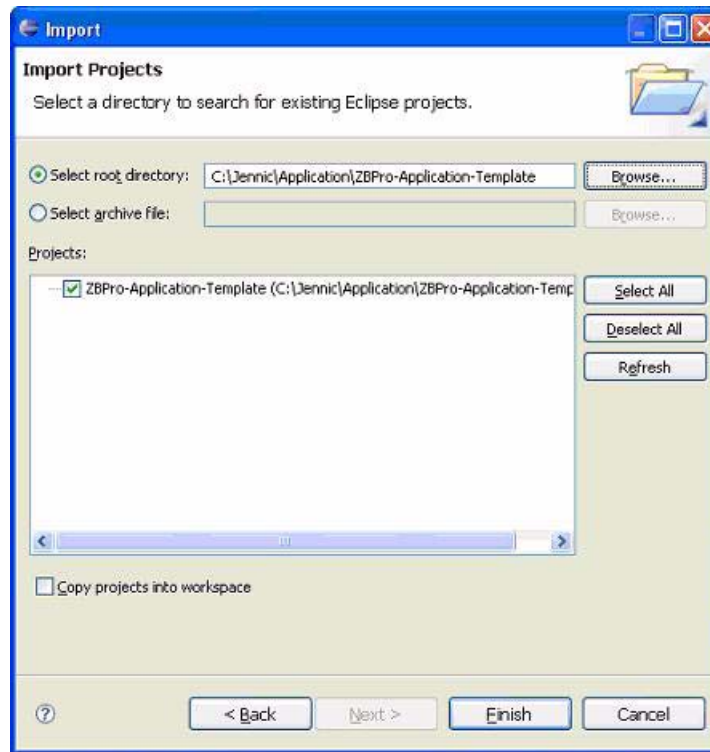
**Step 5** Click **Next** and then navigate down to select your **Application** root folder in the **Browse For Folder** dialogue box.



**Figure 2: 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 3: 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.

**Step 9** Click **OK**.

In the **Project Explorer** pane, your project should now contain six 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 and a 'read me' file which describes how to create a makefile.

**Step 10** 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 B](#). Then edit the associated makefile and add the new source file to the Application Source section as illustrated below:

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

## 2.3 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 4](#). Similarly, click on the **+** symbol next to the **Source** folder.

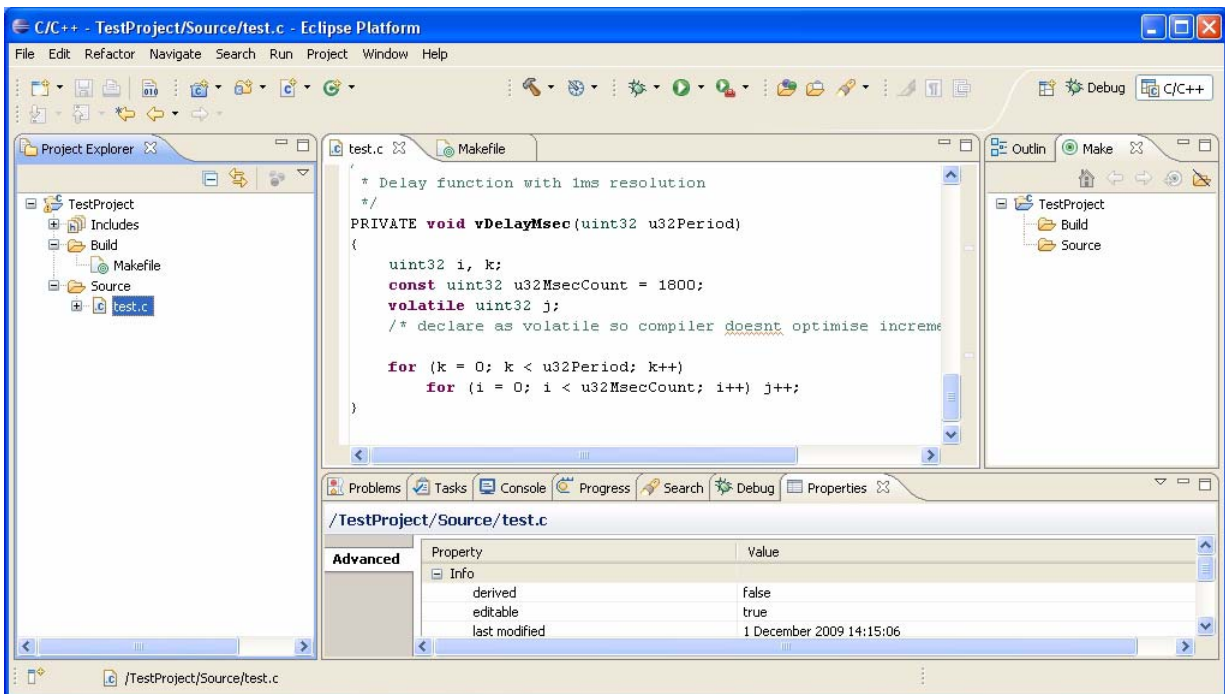


Figure 4: Editing Your C Project

- 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 4](#).
- 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**.

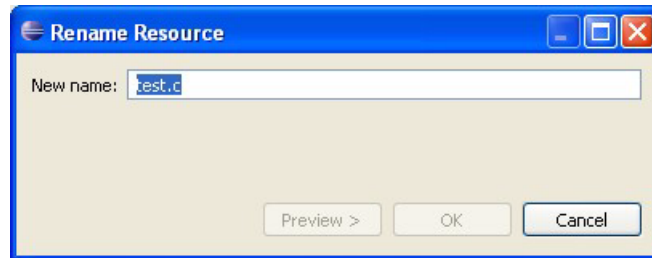


Figure 5: Rename Resource

- 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.

---

## 2.4 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 2.2](#)), and that your editing is complete (see [Section 2.3](#)).
- 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.





---

## 3. Downloading an Application Binary

Once you have built your project, you must download the binary output to the Flash memory of the 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 JN51xx on the target device. Eclipse does not have a built-in Flash programmer, but the Jennic JN51xx Flash Programmer 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 Jennic JN51xx Flash Programmer, refer to the procedure for downloading binary code in the *JN51xx Flash Programmer User Guide (JN-UG-3007)*.

---

### 3.1 Pre-requisites

Ensure that you have the following:

- A target device containing a Jennic wireless 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 the *JN5148 Software Developer's Kit Installation Guide (JN-UG-3064)*.

## 3.2 Download Procedure

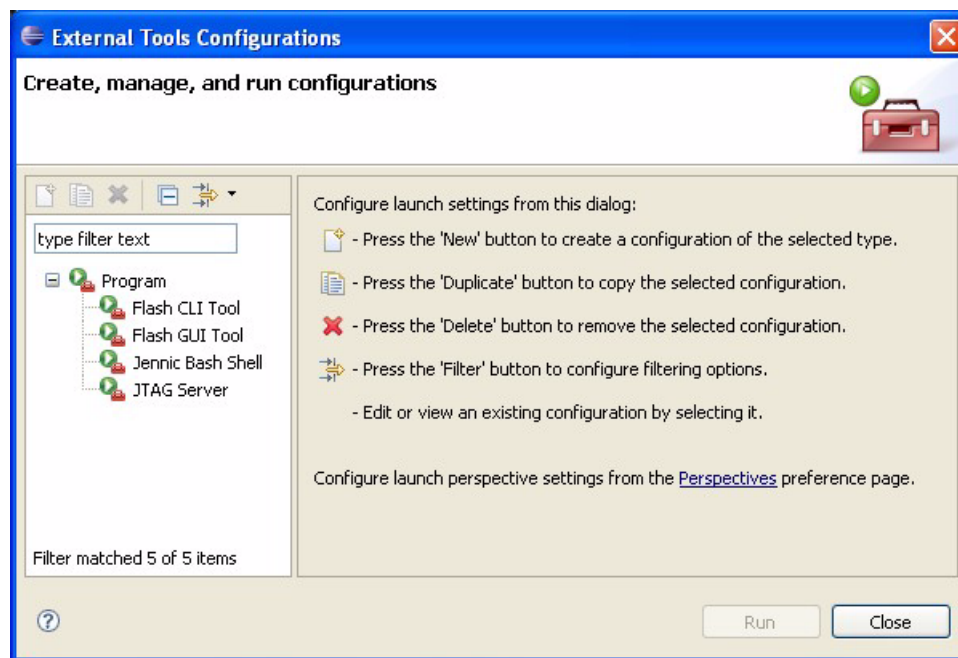
To download your **.bin** file to a device:

- Step 1** Connect a USB port of your PC to the target device using a Jennic-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 C](#).
- 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 6](#).

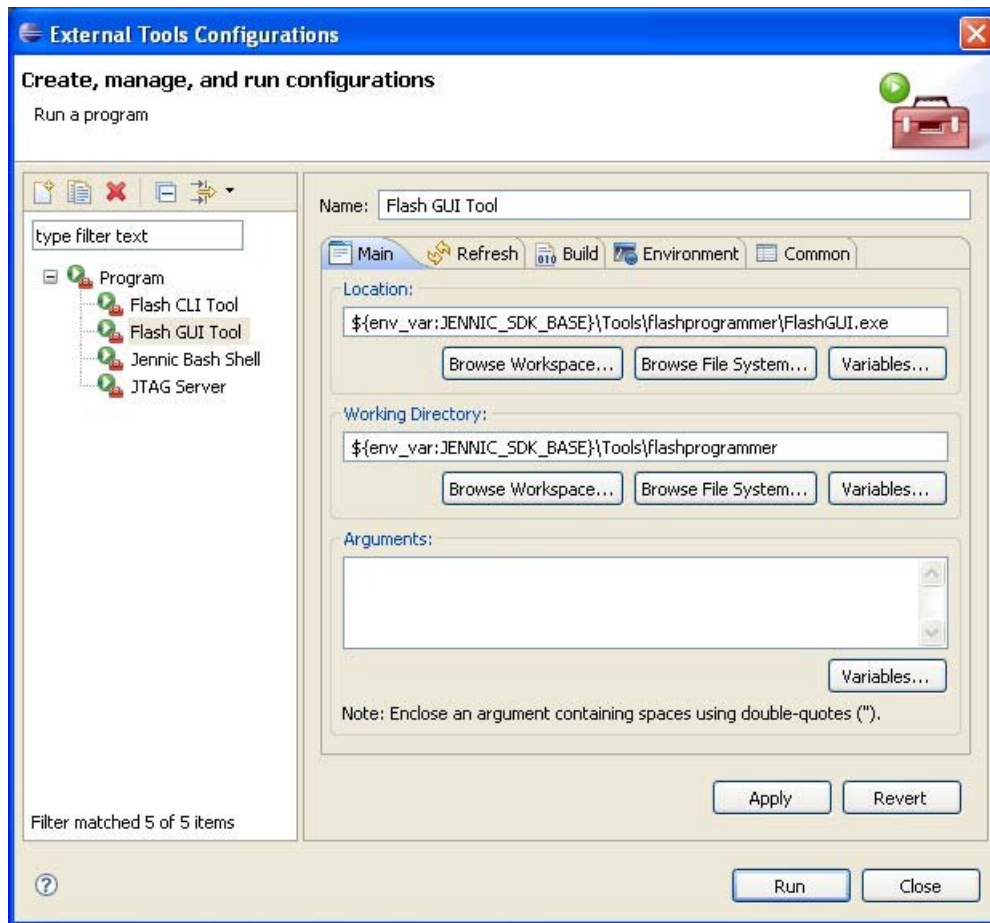


**Note:** You will need to set the PC communications port that has been assigned to the connection to the target device – to identify this port, refer to [Appendix D](#).



**Figure 6: 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 7](#) for the Flash GUI.



**Figure 7: 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 D](#). 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.

---

## 4. 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 2.4](#)).
2. Loaded the resulting binary file into the Flash memory of the device that will run the application (see [Chapter 3](#)).

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 JN51xx-based device (e.g. Jennic 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 JN51xx-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 (in contrast to the method of loading the binary itself into the JN51xx-based device, which is described in the software debug method).

Use of the GDB hardware debugger is described in [Section 4.1](#).

### Using the serial UART to send debugging 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 4.2](#).

Further operational information on the GDB debugger is provided in [Appendix A](#).

## 4.1 GDB Hardware Debugger

### 4.1.1 Configuring the 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 a Jennic-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 C](#).
- 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 C](#).
- 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.

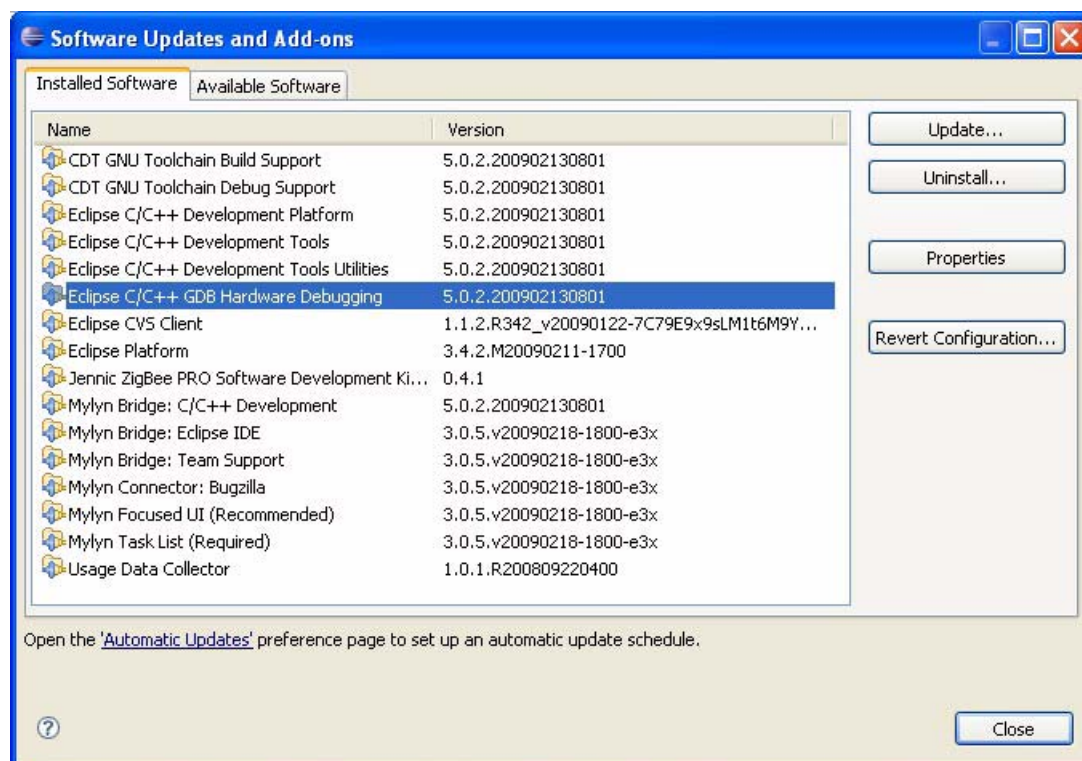


Figure 8: Checking for Hardware Debug Support

- 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**.

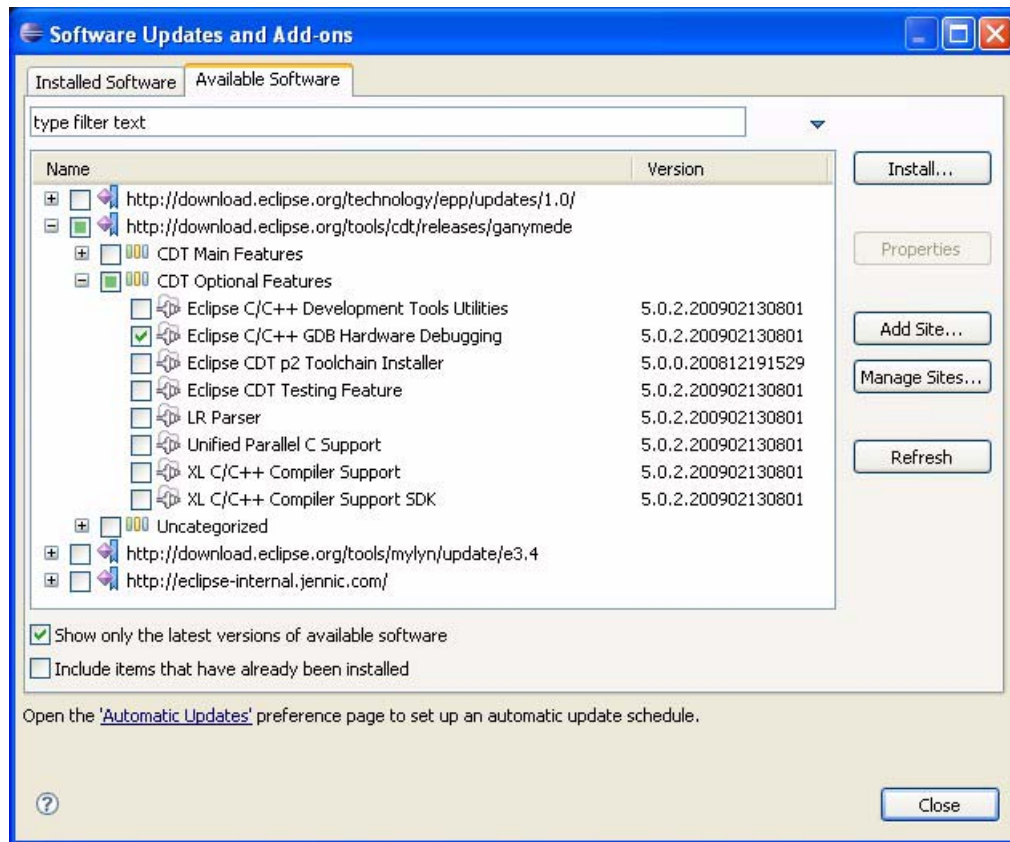
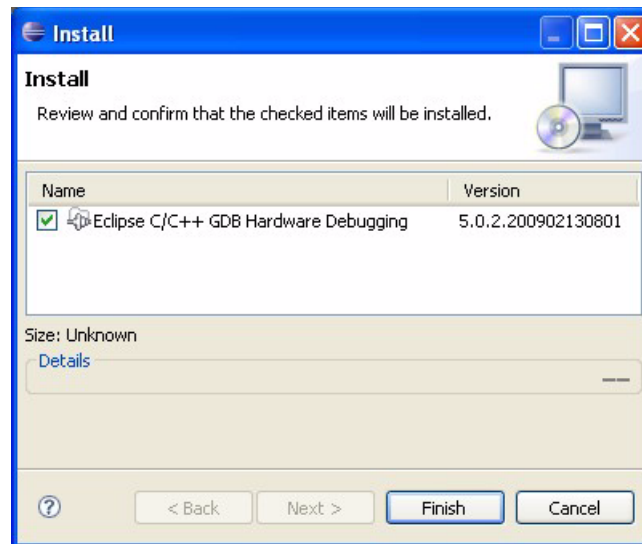


Figure 9: Software Updates Screen

**Step 7** The download program resolves any dependency issues and further requirements, and then presents the **Install** screen (see [Figure 10](#)).



**Figure 10: Software Updates Install Screen**

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

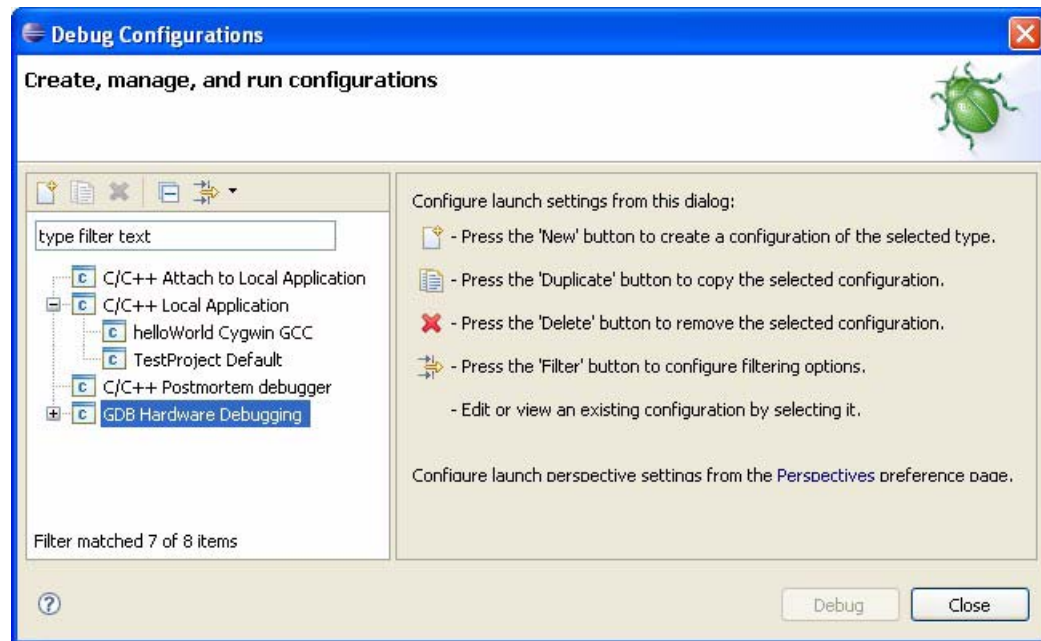
**Step 9** Eclipse now needs to restart in order to incorporate the new plug-ins. Only Eclipse itself will reboot, not the entire machine. Click **Yes** to allow the restart.

**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 11: 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.

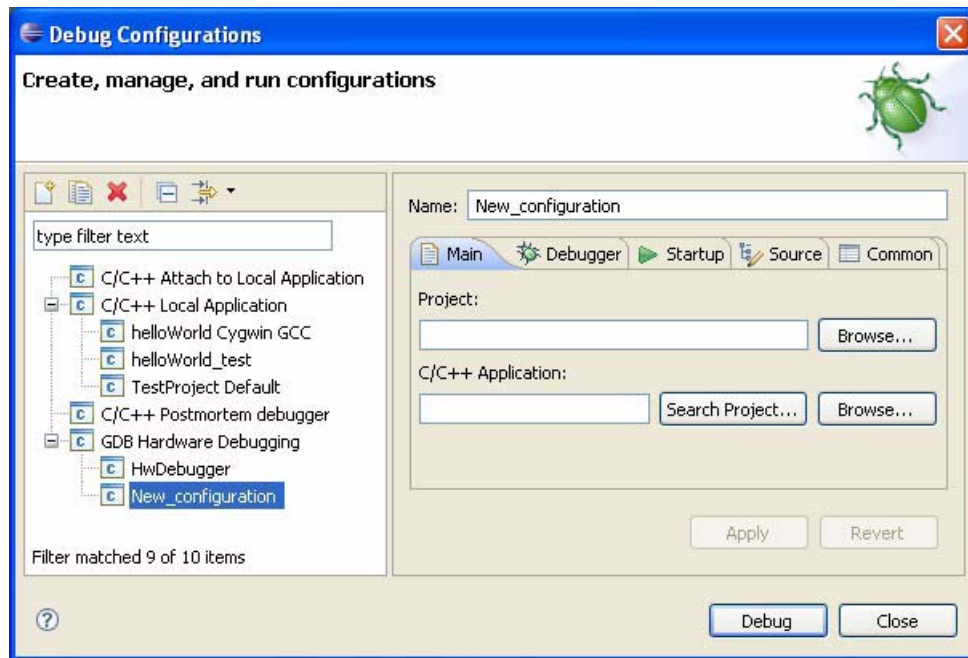
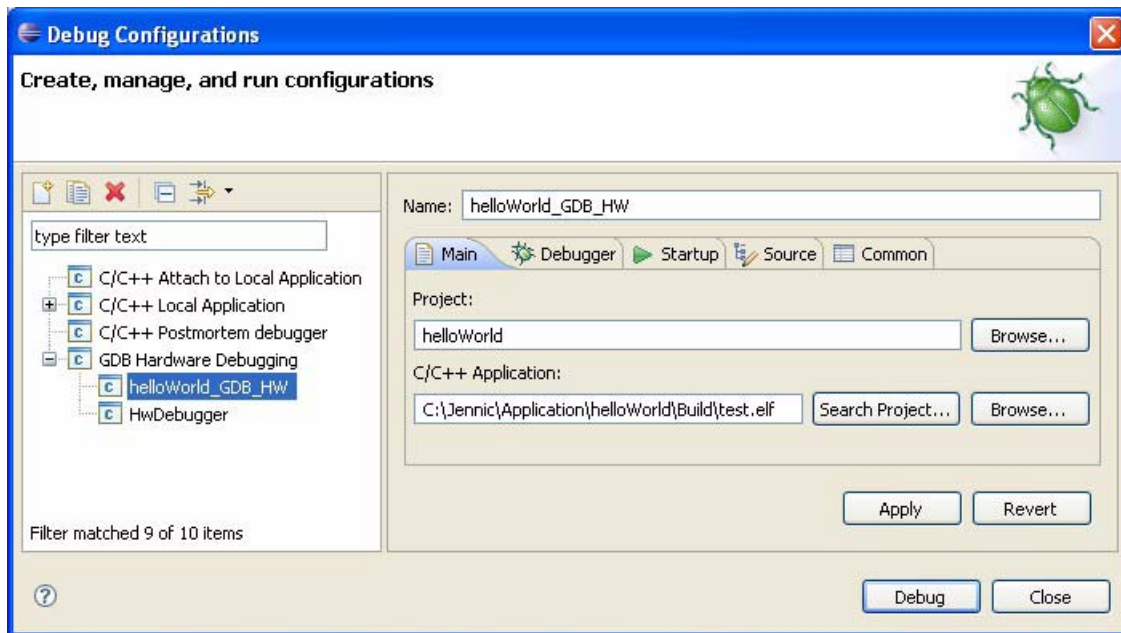


Figure 12: New Hardware Debugger Configuration

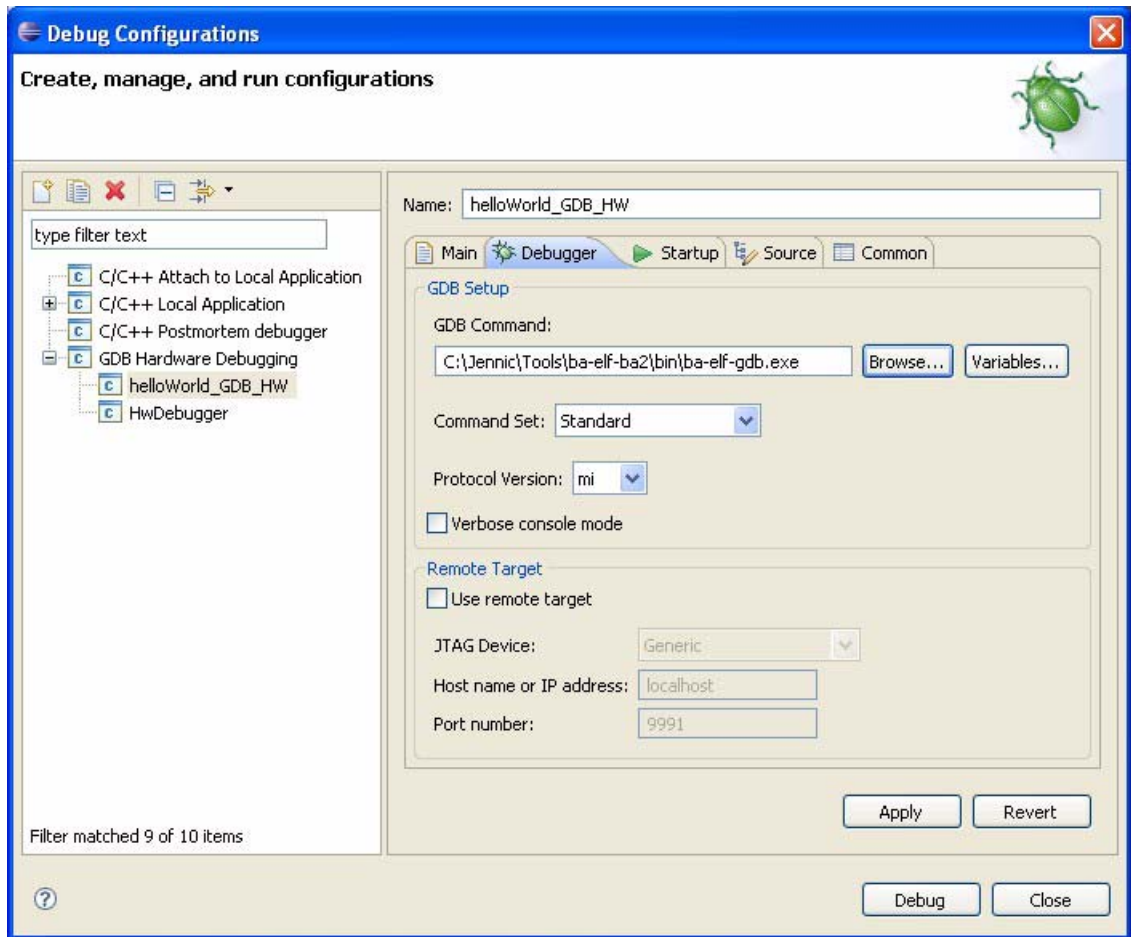
**Step 14** In the **Main** tab, enter the following information (see [Figure 13](#)):

- **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 13: Hardware Debug, Main Tab**

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

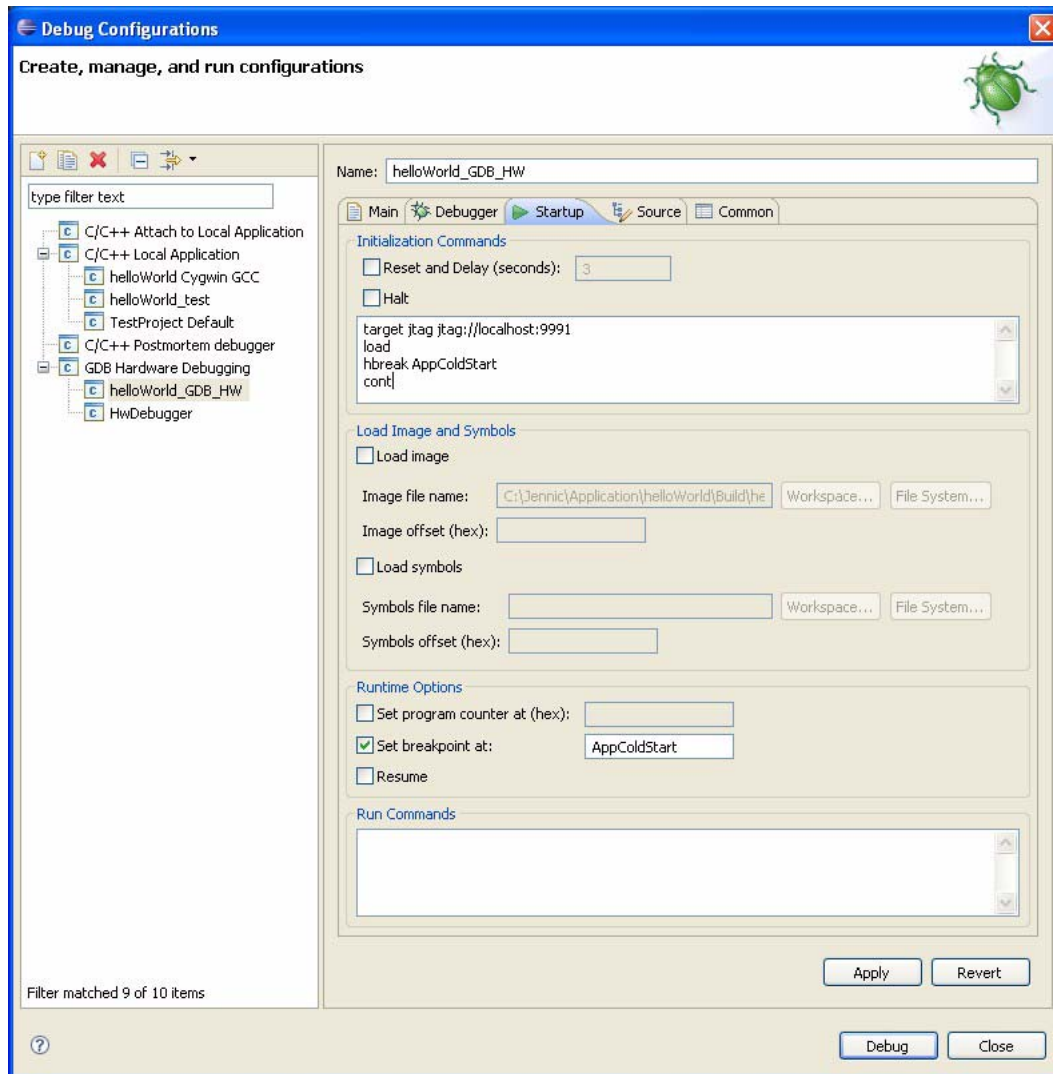


**Figure 14: Hardware Debug, Debugger Tab**

**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.

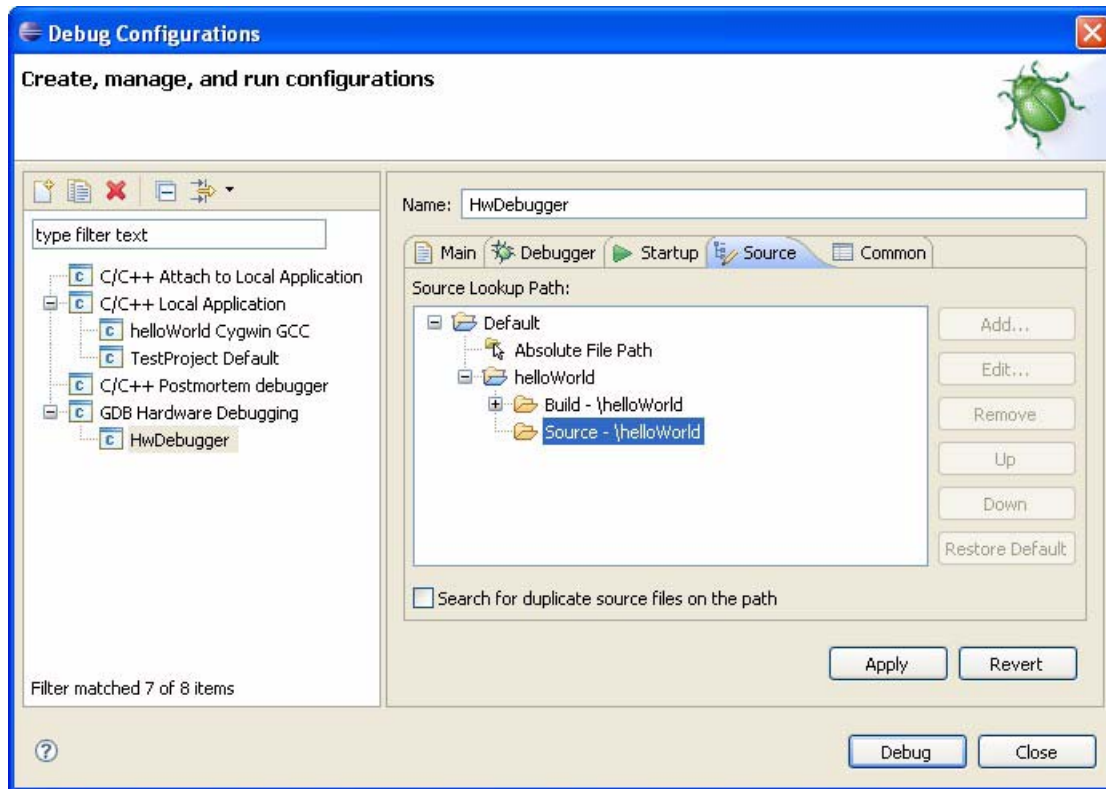


**Figure 15: Hardware Debug, Startup Tab**

**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.

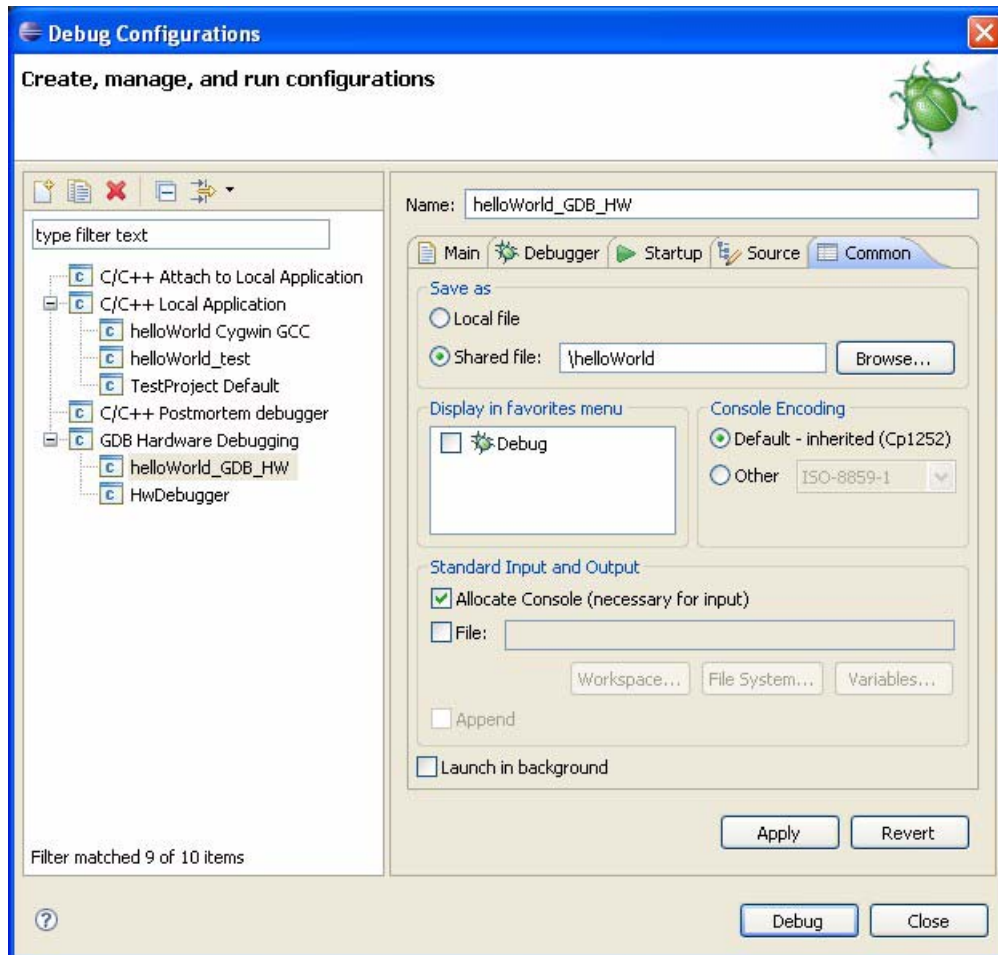
**Step 19** Click **Apply**, then select the **Source** tab.



**Figure 16: 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 17: 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 4.1.2 "Operating the GDB Hardware Debugger"](#).

---

### 4.1.2 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 4.1.1](#).

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 3](#)).

**Step 3** Depending on whether you are using UART0 or UART1 of the JN51xx 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**

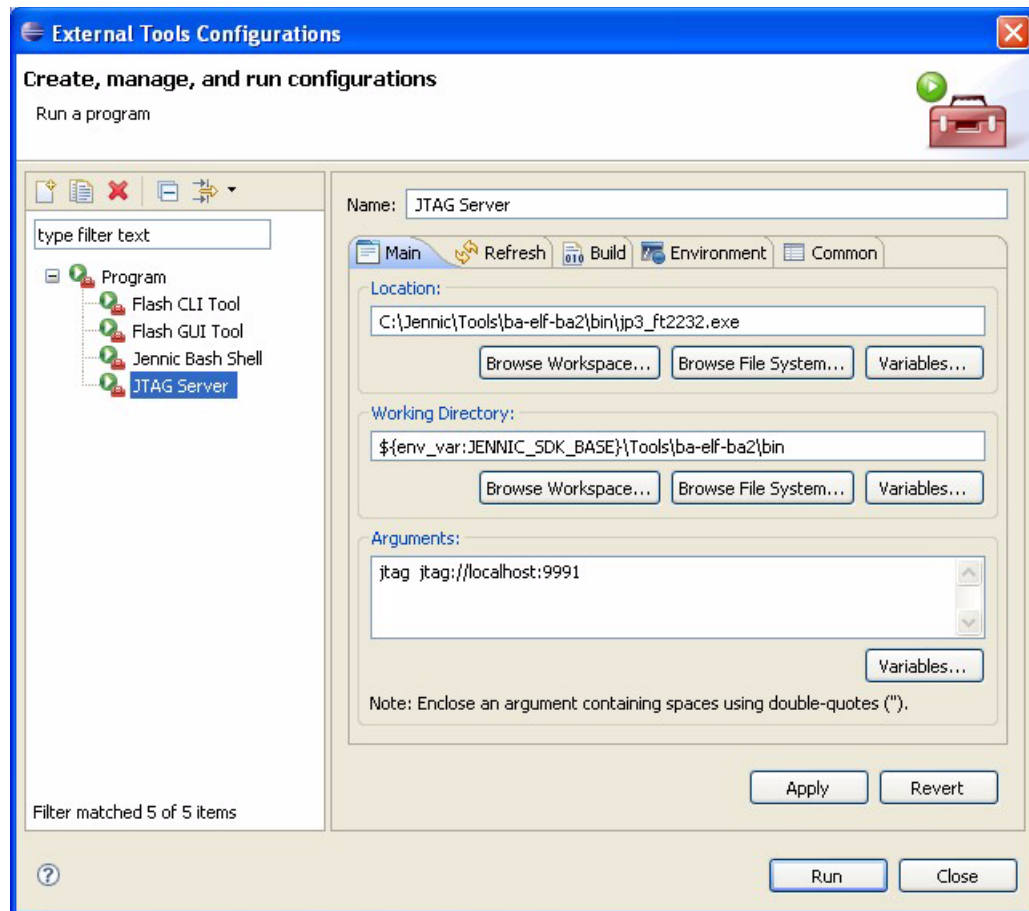
To use the Flash Programmer from within Eclipse, refer to [Chapter 3](#).

**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 18: 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.

---

## 4.2 Real-time Debugging via the Serial Interface

This section describes how to debug real-time network applications using the JN51xx 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.

---

### 4.2.1 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.

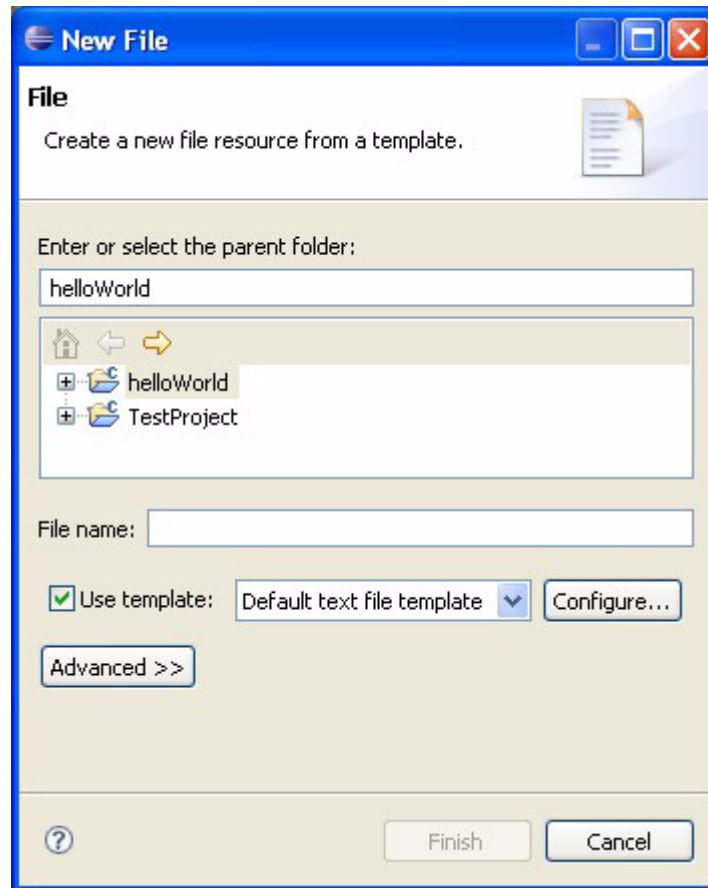


Figure 19: Adding a New File

- 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:

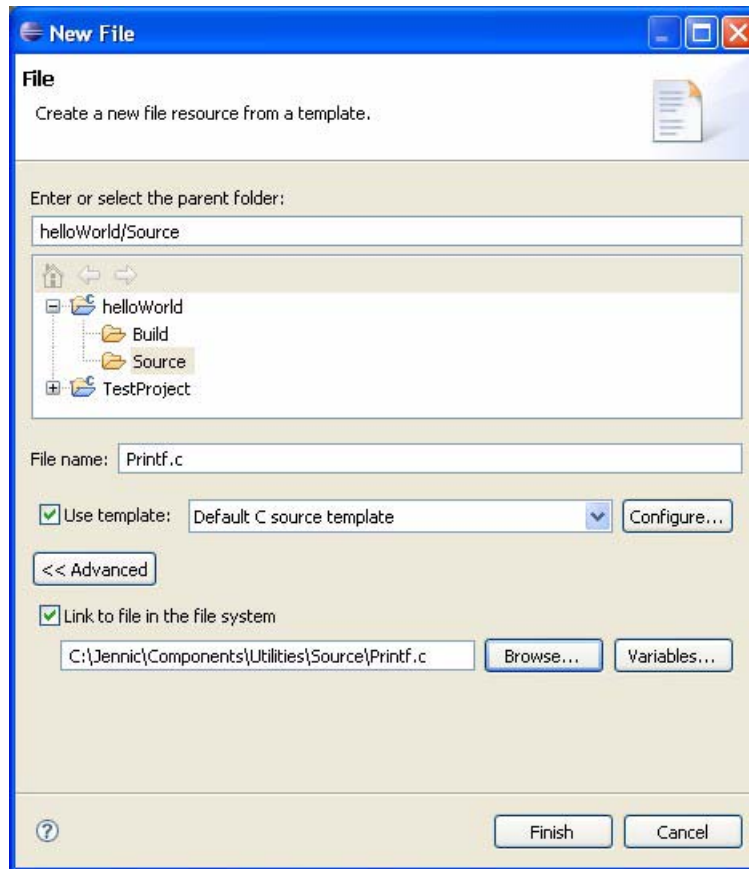


Figure 20: Adding the Printf.c File

**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", ul6NodeId);
```



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

**Step 10** Build the project as described in [Section 2.4](#).

**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](http://www.hw-server.com/software/termv19b.html)).

---

## 4.2.2 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 21: 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 22: 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**.

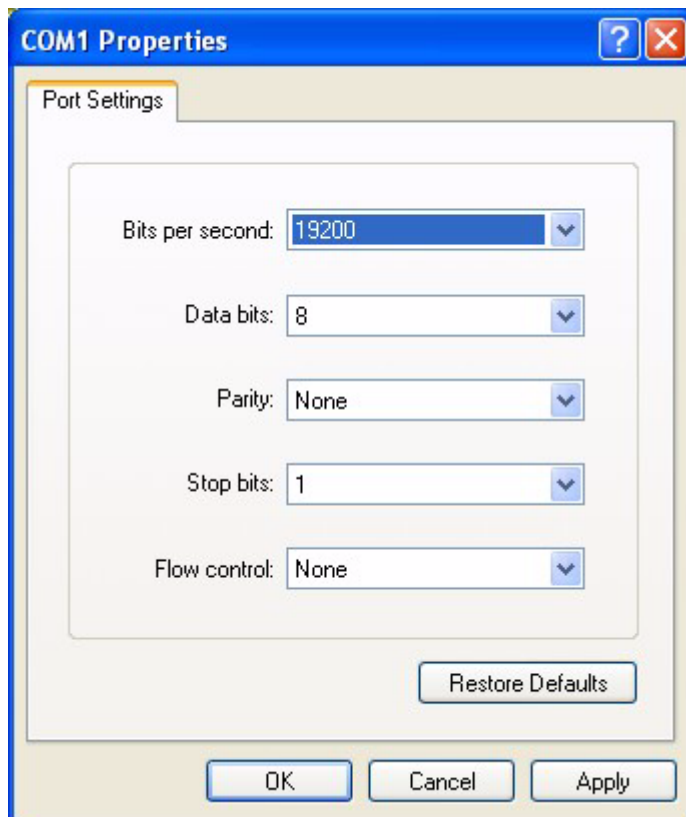


Figure 23: COM Properties Screen

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

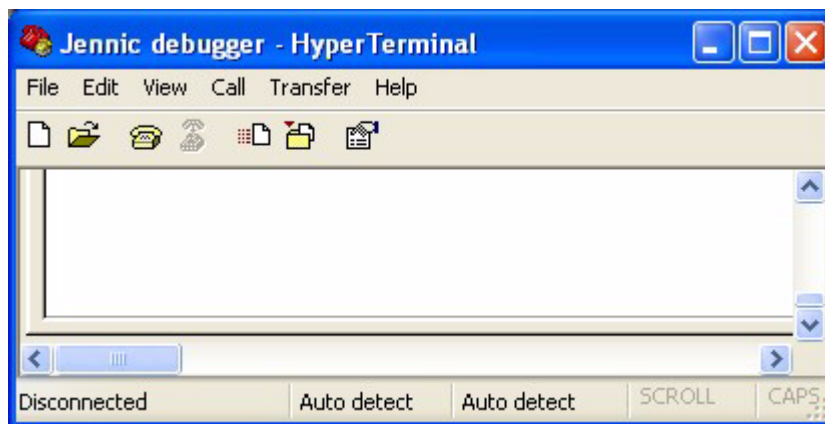


Figure 24: HyperTerminal Screen



---

### 4.2.3 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)*.



---

## Appendices

---

### A. The Integrated Debugger Explained

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 Jennic JN51xx wireless 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 JN51xx 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. Jennic'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 JN51xx?

When a breakpoint is reached, the debug stub is invoked on the JN51xx 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 JN51xx CPU is stalled. *The JN51xx 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 JN51xx?

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.

### Where can I get more information on GDB?

The following are useful web links to further information on GDB:

- For a more detailed description of using GDB in an embedded environment:  
<http://www.embedded.com/1999/9909/9909feat2.htm>
- For an excellent guide to GDB:  
<http://www.dirac.org/linux/GDB/>
- For detailed manuals on GDB:  
<http://docs.freebsd.org/info/GDB/GDB.pdf>  
<http://wwwcdf.pd.infn.it/localdoc/GDBint.pdf>

---

## B. Creating a 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**.

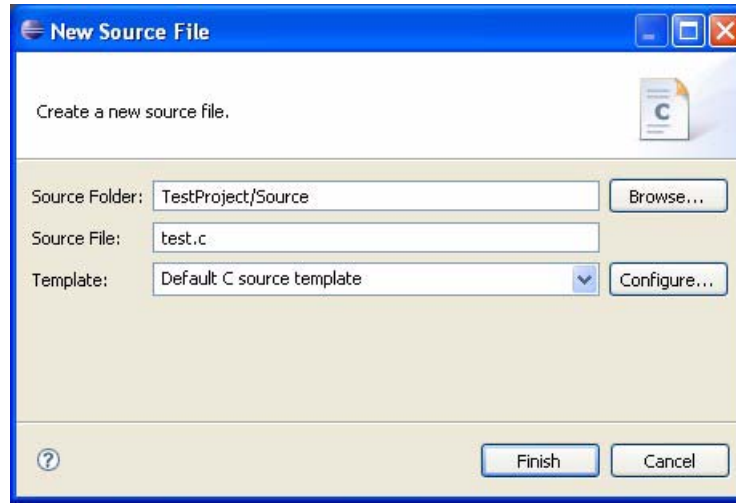


Figure 25: New Source File

- 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.

---

## C. Installing the USB-to-Serial Cable Driver

The USB-to-serial cable supplied with Jennic evaluation and starter kits 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](http://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.

---

## D. 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.

**b)** 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).



**Revision History**

Version	Date	Comments
1.0	7-July-2009	First release
1.1	2-Dec-2009	Added section on creating a build configuration.
1.2	9-Feb-2010	SDK name changed
1.3	20-May-2010	'Creating a project' section modified to be based on importing a Jennic application template. Example "hello world" application removed.

## Important Notice

Jennic reserves the right to make corrections, modifications, enhancements, improvements and other changes to its products and services at any time, and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders, and should verify that such information is current and complete. All products are sold subject to Jennic's terms and conditions of sale, supplied at the time of order acknowledgment. Information relating to device applications, and the like, is intended as suggestion only and may be superseded by updates. It is the customer's responsibility to ensure that their application meets their own specifications. Jennic makes no representation and gives no warranty relating to advice, support or customer product design.

Jennic assumes no responsibility or liability for the use of any of its products, conveys no license or title under any patent, copyright or mask work rights to these products, and makes no representations or warranties that these products are free from patent, copyright or mask work infringement, unless otherwise specified.

Jennic products are not intended for use in life support systems/appliances or any systems where product malfunction can reasonably be expected to result in personal injury, death, severe property damage or environmental damage. Jennic customers using or selling Jennic products for use in such applications do so at their own risk and agree to fully indemnify Jennic for any damages resulting from such use.

All trademarks are the property of their respective owners.

**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](mailto:info@jennic.com)

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

**www.Jennic.com**  
TECHNOLOGY FOR A CHANGING WORLD