



# **BeyondStudio for NXP Installation and User Guide**

JN-UG-3098  
Revision 1.2  
13 March 2015



---

# Contents

<b>Preface</b>	<b>5</b>
Organisation	5
Conventions	6
Acronyms and Abbreviations	6
Related Documents	6
Support Resources	7
Trademarks	7
Acknowledgements	7
<b>1. Introduction and Installation</b>	<b>9</b>
1.1 Features	9
1.2 Installation Procedure	10
1.2.1 Installing BeyondStudio for NXP	11
1.2.2 Installing the JN516x SDK	13
1.2.3 Installing the ZigBee Plug-ins	14
1.3 Directory Structure	18
1.4 Basic Workflow	19
1.5 Starting BeyondStudio for NXP	20
1.6 Workbench Perspectives and Layout	21
<b>2. Creating/Importing a Project</b>	<b>23</b>
2.1 Creating a Workspace	24
2.2 Importing a Project	26
2.3 Adding a New Source File	28
<b>3. Developing an Application</b>	<b>29</b>
3.1 Working on a Project	29
3.2 Building a Project	30
3.3 Loading an Application into a Device	31
3.4 Creating and Using a Run Configuration	34
<b>4. Debugging an Application</b>	<b>37</b>
4.1 Creating a Debug Configuration	37
4.2 Connecting PC to JN516x Device for Debugging	38
4.3 Launching the Debugger	41
4.4 Debug Perspective	42

4.5 Debug Operation Summary	43
<b>Appendices</b>	<b>45</b>
<b>A. Identifying the PC Communications Port Used</b>	<b>45</b>
<b>B. Using a Workspace Outside the Installation Folder</b>	<b>45</b>
<b>C. Eclipse Enhancements</b>	<b>46</b>
C.1 Integrated Serial Terminal	46
C.2 Run Configurations	46
C.3 Flash Programmer	46
C.4 Flash Memory and EEPROM Dump	46
<b>D. Creating and Using Serial Terminals</b>	<b>47</b>
D.1 Creating/Enabling a Terminal Tab	47
D.2 Including a Terminal Tab in a Run Configuration	49
<b>E. Reading the Contents of Flash Memory and EEPROM</b>	<b>50</b>

---

## Preface

Welcome to the *BeyondStudio for NXP Installation and User Guide*. This manual describes the installation and operation of the BeyondStudio IDE (Integrated Development Environment) which has been adapted for use with the NXP JN516x family of wireless microcontrollers. BeyondStudio for NXP is supplied in the software package JN-SW-4141, available via the NXP Wireless Connectivity TechZone.



**Note 1:** BeyondStudio for NXP is currently available only for certain wireless network protocol stacks, as indicated on the relevant tabs of the TechZone.

**Note 2:** For support with BeyondStudio for NXP, you should consult this manual and not the BeyondStudio Manual produced by Beyond Semiconductor.

---

## Organisation

This manual consists of 4 chapters and 5 appendices, as follows:

- [Chapter 1](#) introduces BeyondStudio for NXP and provides installation instructions for this and NXP's JN516x SDKs, as well as other essential information
- [Chapter 2](#) describes how to create a project in BeyondStudio for NXP by importing an existing project from the NXP Application Note on which the new project will be based
- [Chapter 3](#) describes how to use BeyondStudio for NXP to develop, build and load a JN516x application
- [Chapter 4](#) describes how to use BeyondStudio for NXP to debug a JN516x application
- The [Appendices](#) describe:
  - finding the PC serial communications port that has been allocated to a USB connection
  - using a workspace outside of the BeyondStudio for NXP directory
  - the Eclipse enhancements in this release
  - setting up a terminal emulator in BeyondStudio for NXP
  - reading the contents of JN516x non-volatile memory

---

## 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

IDE	Integrated Development Environment
SDK	Software Developer's Kit

---

## Related Documents

JN-DS-JN516x	JN516x Data Sheet
JN-AN-1202	BeyondStudio Migration Guidelines Application Note
JN-AN-1203	JN516x JTAG Debugging in BeyondStudio Application Note
JN-UG-3093	JN516x-EK001 Evaluation Kit User Guide
JN-UG-3087	JN516x Integrated Peripherals API User Guide
JN-UG-3101	ZigBee PRO Stack User Guide
JN-UG-3075	JenOS User Guide

---

## Support Resources

To access online support resources such as SDKs, Application Notes and User Guides, visit the Wireless Connectivity TechZone:

[www.nxp.com/techzones/wireless-connectivity](http://www.nxp.com/techzones/wireless-connectivity)

All NXP resources referred to in this manual can be found at the above address, unless otherwise stated.

---

## Trademarks

All trademarks are the property of their respective owners.

---

## Acknowledgements

BeyondStudio is a product of Beyond Semiconductor, based on the open-source Eclipse IDE. BeyondStudio for NXP is an adaptation produced by NXP for the JN516x wireless microcontrollers under a technical licence from Beyond Semiconductor. BeyondStudio for NXP is based on BeyondStudio v1.1.9.

All support requests for BeyondStudio for NXP should be directed to NXP.





# 1. Introduction and Installation

The “BeyondStudio for NXP” Integrated Development Environment (IDE) provides a platform for the development of wireless network applications to be run on NXP’s JN516x family of wireless microcontrollers. It is supplied in the NXP software package JN-SW-4141 and contains the complete toolchain required for JN516x application development on a PC.



**Note 1:** BeyondStudio for NXP is currently available only for use with certain wireless network protocol stacks. The required toolchain for a given protocol stack is indicated on the relevant tabs of the NXP Wireless Connectivity TechZone. Be sure to use the correct toolchain for your chosen protocol.

**Note 2:** BeyondStudio for NXP must be installed before the JN516x Software Developer’s Kit (SDK) for your chosen protocol. Full installation instructions for both components are provided in [Section 1.2](#).

**Note 3:** All support issues concerning BeyondStudio for NXP should be addressed to NXP.

## 1.1 Features

BeyondStudio for NXP is an NXP-adaptation of the BeyondStudio IDE (Integrated Development Environment) v1.1.9, produced by Beyond Semiconductor, based on the Kepler release of the open-source Eclipse IDE.

The main features of BeyondStudio for NXP are:

- Source code editor with functionality such as syntax highlighting, custom code style and code cross-referencing
- GUI-based configuration
- Graphical source code debugger using JTAG
- Integrated GNU-based cross-compiling toolchain, including GCC 4.7.3, binutils 2.22 and GDB 7.5.1 (the toolchain can also be used directly from a command line window)
- Integrated JN516x Flash programmer for loading compiled binary files into JN516x Flash memory (internal)
- Integrated serial console
- Extendable functionality through plug-ins

BeyondStudio for NXP can be used with the 32-bit and 64-bit editions of Microsoft Windows 7 and 8.

---

## 1.2 Installation Procedure

This section describes how to install both the BeyondStudio for NXP (JN-SW-4141) and the JN516x SDK for the desired wireless network protocol stack. Before proceeding with the installation, note the following:

- BeyondStudio for NXP must be installed before the chosen JN516x SDK
- Currently, BeyondStudio for NXP can be used only with the JN516x SDKs for certain wireless network protocol stacks, as indicated in the information for the different protocol stacks on the NXP Wireless Connectivity TechZone. The relevant SDKs have part numbers of the form JN-SW-41xx
- The NXP JN516x SDKs for the ZigBee profiles (e.g. Home Automation) include Eclipse-based plug-ins which allow the ZPS Configuration Editor and JenOS Configuration Editor to be used within BeyondStudio for NXP
- BeyondStudio for NXP requires up to 700 MBytes of hard-disk space and the JN516x SDK will require additional disk space, depending on the SDK type
- A single installation of BeyondStudio for NXP can be shared between multiple (valid) JN516x SDKs or a dedicated instance can be installed for each SDK (in different locations)
- If replacing a previous installation of BeyondStudio for NXP and the existing SDK directory structure contains custom files, you must back up these files before starting the installation, since the directory tree of the existing SDK installation will be automatically removed and replaced
- If you have a previous JN51xx SDK Toolchain (e.g. JN-SW-4041) on your PC, you do not need to uninstall it before installing BeyondStudio for NXP

For further information about the software to be installed, refer to the Release Notes packaged with the software.

Installation instructions are provided in the sub-sections below and must be performed in the following order:

1. Install BeyondStudio for NXP as described in [Section 1.2.1](#).
2. Install your chosen JN516x SDK as described in [Section 1.2.2](#).
3. If you are using a ZigBee-based SDK, install the configuration editor plug-ins into BeyondStudio for NXP as described in [Section 1.2.3](#).

## 1.2.1 Installing BeyondStudio for NXP

BeyondStudio for NXP is provided as the following file:

### **JN-SW-4141 BeyondStudio for NXP.exe**

This installer is available from the NXP Wireless Connectivity TechZone. It is supplied in a ZIP file which also contains the installer's Release Notes.

During the installation, the following components can be selected:

- BeyondStudio for NXP (mandatory)
- JN51xx Compiler Tools (mandatory)
- Java JRE (optional)
- MSys (optional)
- Zadig (optional)

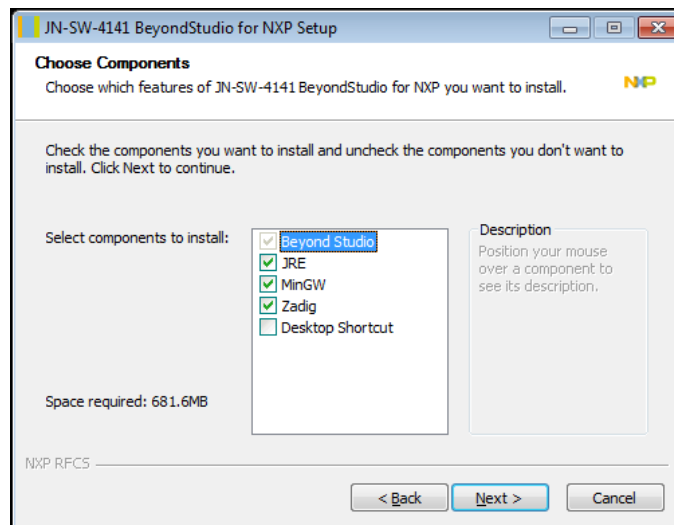
If any one of JRE, MSys and Zadig already exists on the installation computer, you can choose not to install it.

You are advised to install BeyondStudio for NXP into the following directory:

**C:\NXP\bstudio\_nxp**

The installation procedure is described below.

- Step 1** Launch the installer **JN-SW-4141 BeyondStudio for NXP.exe** on your machine. The setup wizard will start.
- Step 2** Follow the on-screen instructions of the set-up wizard until you reach the **Choose Components** screen:



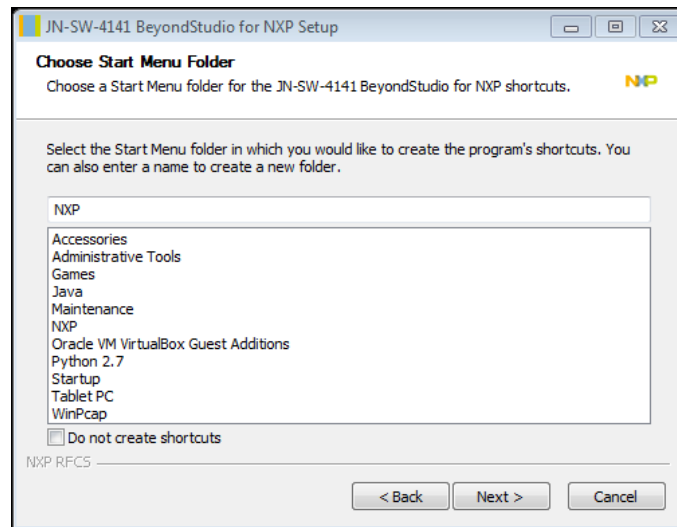
By default, all the components are selected. De-select any component(s) that you do not wish to install. BeyondStudio for NXP and the JN51xx Compiler Tools are mandatory. If any one of JRE, MSys and Zadig already exists on the computer, you may choose not to install it.

## Chapter 1

### Introduction and Installation

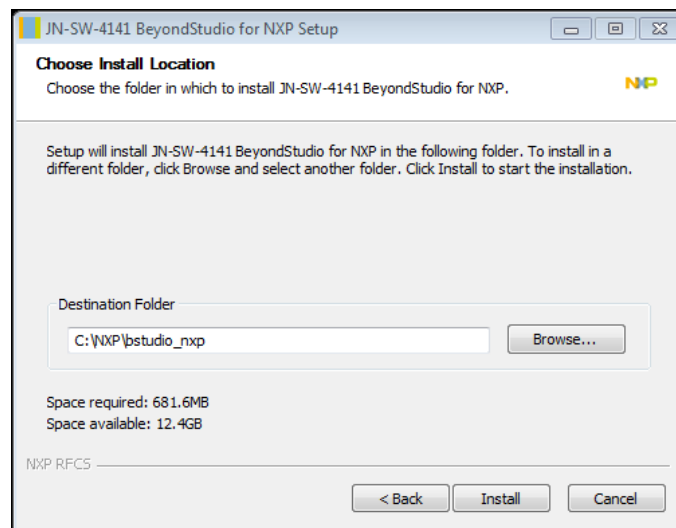
Click **Next** to continue.

**Step 3** In the next screen, specify the folder in which you want BeyondStudio for NXP to appear in the Windows **Start** menu. By default, this is set to **NXP**.



Click **Next** to continue.

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



The set-up wizard will automatically insert the installation directory. By default, this is **C:\NXP\bstudio\_nxp**. If required, you can specify another drive/path.

Click **Install**.

**Step 5** Wait for the installation to complete (this may take several minutes) and then click **Next** followed by **Finish**.



**Note:** Before starting the installation, the installer will try to detect an existing installation in the specified directory. If one is found, you will be asked to confirm that the installer can remove this directory before proceeding.

**Step 6** Now install the required JN516x SDK, as described in [Section 1.2.2](#).



**Note:** In order to use the Flash programming functionality of BeyondStudio for NXP, the device driver for an FTDI USB-to-serial chip is required on your PC. If this device driver is not already present, you can install it at any time before Flash programming is required. Obtain the driver from the VCP drivers page of the FTDI web site: [www.ftdichip.com/Drivers/VCP.htm](http://www.ftdichip.com/Drivers/VCP.htm). To perform the installation, a device or cable containing an FTDI chip must be connected to a USB port of your PC.

---

## 1.2.2 Installing the JN516x SDK

Once BeyondStudio for NXP has been installed as described in [Section 1.2.1](#), the JN516x SDK for the desired wireless network protocol stack can be installed. The contents of the SDK are indicated in the User Guide for the relevant protocol, which is available from the NXP Wireless Connectivity TechZone.



**Note:** Currently, BeyondStudio for NXP can be used only with the JN516x SDKs for certain wireless network protocol stacks, as indicated in the information for the different protocol stacks on the NXP Wireless Connectivity TechZone. The relevant SDKs have part numbers of the form JN-SW-41xx.

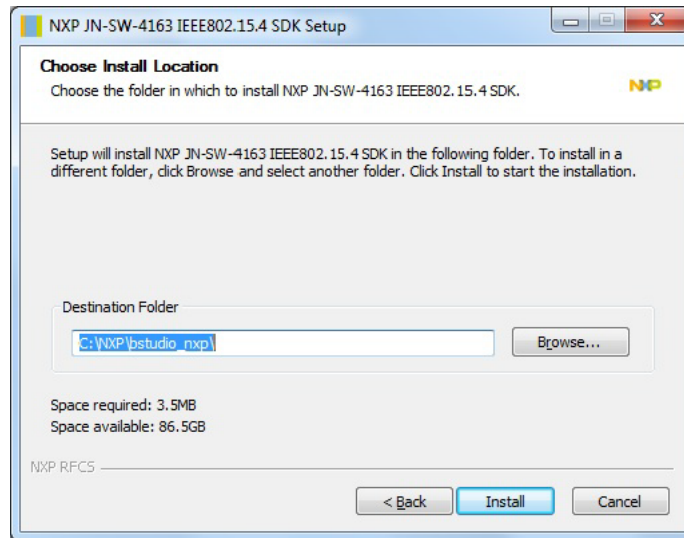
The installation procedure is described below (the screenshots show IEEE 802.15.4 as an example).

**Step 1** Ensure that you have installed BeyondStudio for NXP, as described in [Section 1.2.1](#).

**Step 2** Launch the relevant JN516x SDK installer on your machine. The SDK 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 - this should match the installation directory specified in [Section 1.2.1](#) for BeyondStudio:



The set-up wizard will automatically insert the installation directory. By default, this is **C:\NXP\bstudio\_nxp**. If required, you can specify another drive/path.

Click **Install**.

- Step 5** Wait for the installation to complete and then click **Finish**.

The directory structure created on your hard disk is illustrated in [Section 1.3](#).

---

## 1.2.3 Installing the ZigBee Plug-ins

If you have installed a JN516x SDK for a ZigBee profile (e.g. Home Automation) then you will now need to install the configuration editor plug-ins into BeyondStudio for NXP. These plug-ins are provided in the JN516x SDK for the ZigBee profile and are required for developing ZigBee PRO applications. They are:

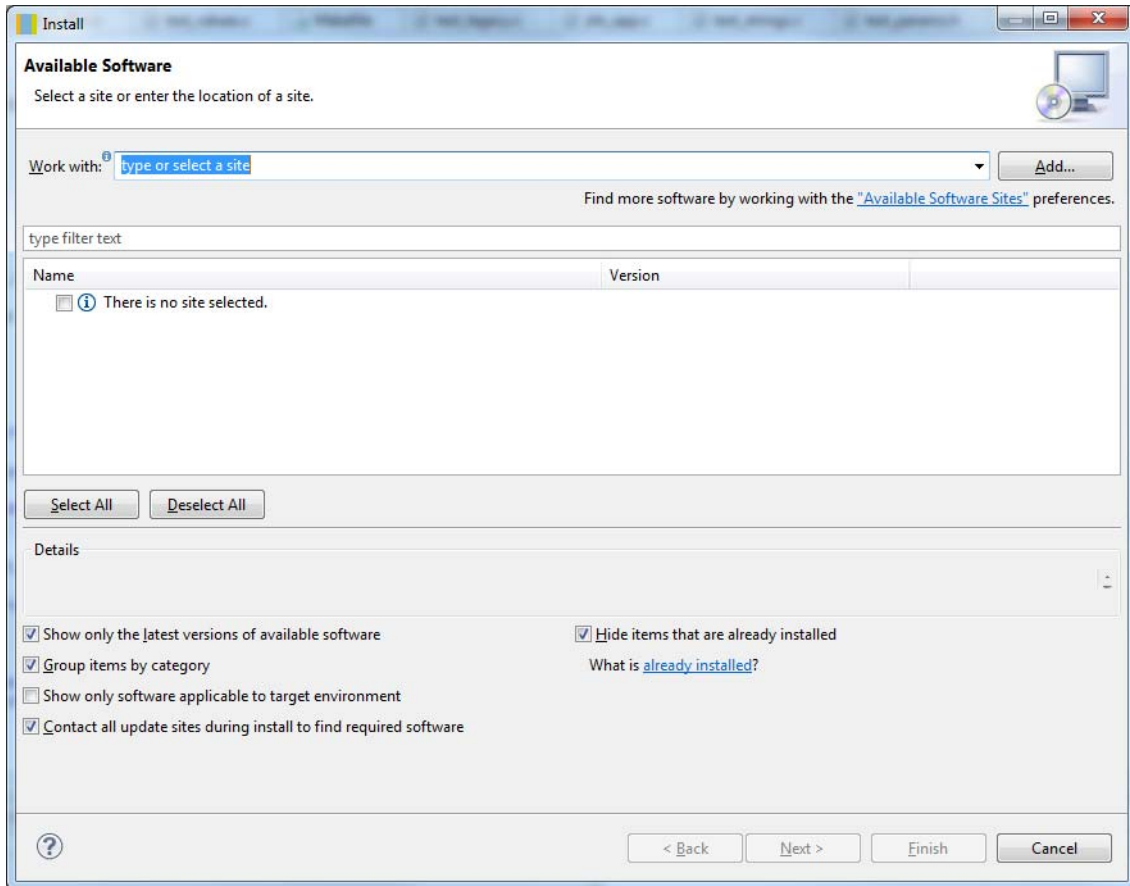
- **ZPS Configuration Editor:** This editor provides a convenient way to set ZigBee network parameters, such as the properties of the Co-ordinator, Routers and End Devices (for example, by setting elements of the device descriptors). For more information on this editor, refer to the *ZigBee PRO Stack User Guide (JN-UG-3101)*.
- **JenOS Configuration Editor:** This editor provides a graphical interface for configuring the way an application uses JenOS resources, such as timers, mutexes and Interrupt Service Routines. For more information on this editor, refer to the *JenOS User Guide (JN-UG-3075)*.



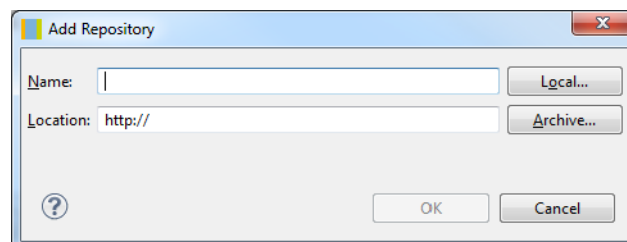
**Note:** Building an application requires the configuration tool command line utilities, which are provided in the JN516x SDK installer and were installed as part of the procedure that you followed in [Section 1.2.2](#).

The installation procedure for the plug-ins is described below.

- Step 1** Start BeyondStudio for NXP (if not already started) and ensure that your installation machine is connected to the Internet (this connection is needed in order to download the required Eclipse dependencies).
- Step 2** In the BeyondStudio main menu, follow the path **Help > Install New Software...**. The following **Available Software** screen will be displayed.



- Step 3** In the **Available Software** screen, click the **Add...** button. The following **Add Repository** window will be displayed.





**Step 4** In the **Add Repository** window:

- a) Click the **Local...** button and browse to the plug-ins' sub-folder of the installed SDK - for example:

**C:\NXP\bstudio\_nxp\sdk\JN-SW-4168\Tools\Eclipse\_plugins\com.nxp.sdk.update\_site**

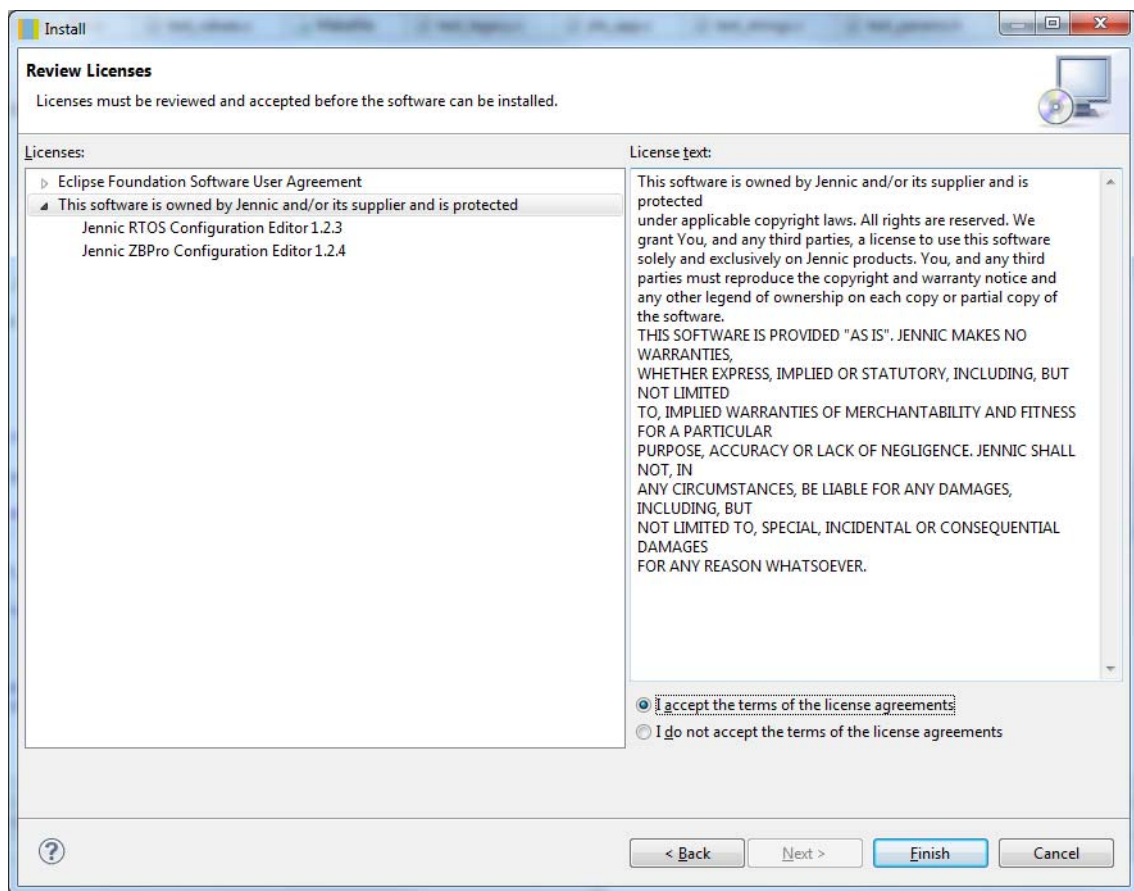
- b) In the **Name** field, enter a name for the plug-ins repository (e.g. ZigBee Plugins).

- c) Click **OK**. The new repository will be available in the **Available Software** screen.

**Step 5** In the **Available Software** screen, expand **Jennic ZBPro SDK** and ensure that the checkboxes for **Jennic RTOS Configuration Editor** and **Jennic ZBPro Configuration Editor** are ticked. Click **Next >** to continue.

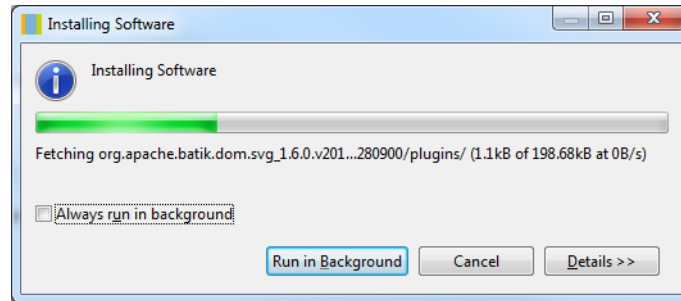
**Step 6** Wait for 'calculating requirements and dependencies' to complete (this may take a few minutes) and when the **Install Details** window appears, click **Next >**.

**Step 7** Read the terms of the license agreements displayed on the **Review Licenses** screen (see below) and select the appropriate radio button.

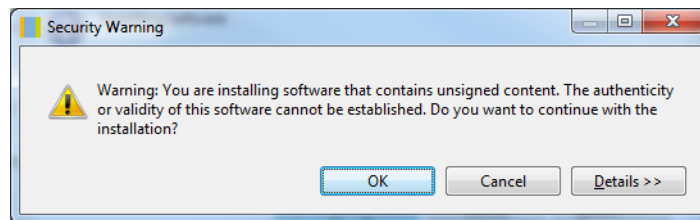




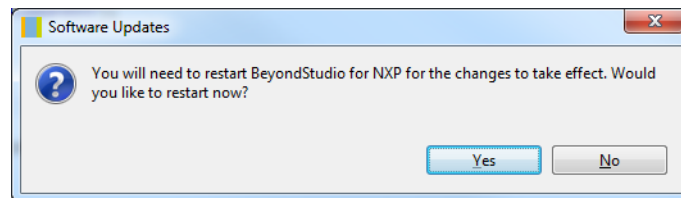
**Step 8** Click **Finish** to continue. Assuming you have accepted the license agreements, BeyondStudio installs the plug-ins. If desired, click the button for Run in Background.



**Step 9** During the installation process, a screen may appear to ask you to accept the installation of unsigned content. If you agree, click **OK** to continue.



**Step 10** BeyondStudio now needs to restart in order to incorporate the new plug-ins (only BeyondStudio itself will reboot, not the entire machine). Click **Yes** to allow the restart.



**Step 11** You are now returned to the BeyondStudio main screen.

For information on the configuration editors, refer to the:

- *ZigBee PRO Stack User Guide (JN-UG-3101)* for the ZPS Configuration Editor
- *JenOS User Guide (JN-UG-3075)* for the JenOS Configuration Editor

## 1.3 Directory Structure

During installation, the following directory structure is created and populated.

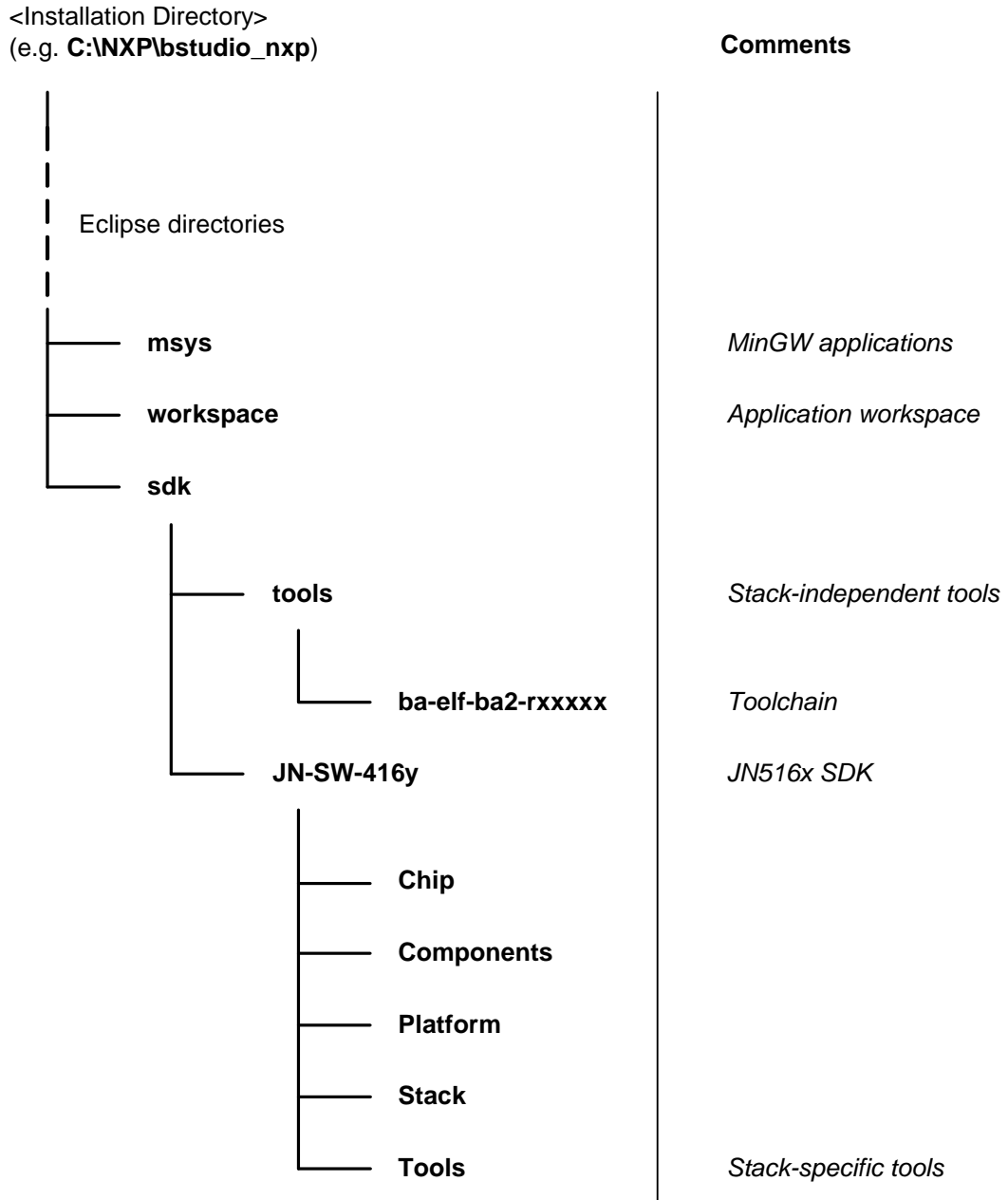


Figure 1: Directory Structure

## 1.4 Basic Workflow

This section outlines the main workflow stages of application development in BeyondStudio for NXP that are detailed in the remaining chapters of this manual. These stages are as follows:

### Stage 1: Create/Import a project

In BeyondStudio, an application under development is termed a project. Therefore, a project must first be created. For JN516x application development, NXP provide Application Notes containing template and example applications that can be used as a starting point for custom application development. These applications are supplied with project files. You should therefore import a project from the relevant NXP Application Note rather than create a new project from scratch. Importing a project into BeyondStudio is detailed in [Chapter 2](#). You can then write/edit your application code within the source code editor of BeyondStudio.

### Stage 2: Build, load and run the application

Once you have written your application code, you can build the application binary using the JN51xx compiler which is integrated into BeyondStudio for NXP. You can then use the integrated JN51xx Flash programmer to load the binary file into the internal Flash memory of a JN516x wireless microcontroller. Building an application and loading it into JN516x Flash memory are described in [Chapter 3](#).



**Tip:** BeyondStudio for NXP provides a 'Run Configuration' feature which allows one or more applications to be built, loaded into the target device(s) and run as the result of a single click of a button.

### Stage 3: Debug the application

Application development normally involves a debug stage. This can be conducted from BeyondStudio, which interfaces to the application running on a JN516x device via a JTAG interface. Debugging a running application using BeyondStudio for NXP is described in [Chapter 4](#).

## 1.5 Starting BeyondStudio for NXP

BeyondStudio for NXP can be started either via the Windows **Start** menu or by double-clicking on the desktop shortcut (if set up).

On starting BeyondStudio for the first time, you will be asked to specify a 'workspace' directory in the **Workspace Launcher** dialogue box (see [Figure 2](#) below). This directory is a space on your computer where your project(s) will be kept and sits in the BeyondStudio for NXP directory structure as indicated in [Section 1.3](#).

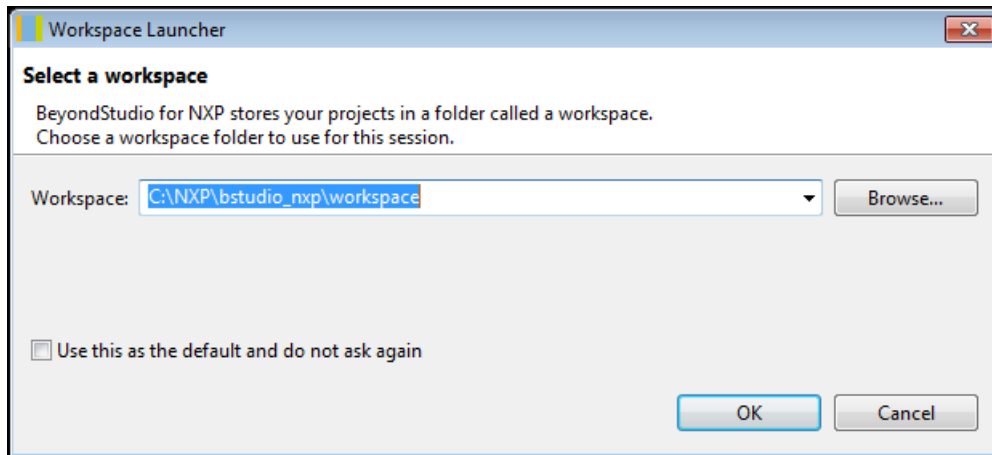


Figure 2: Workspace Launcher Dialogue Box

- You can use a single workspace for all your projects. In this case, if you tick the “Use this as the default and do not ask again” checkbox in the **Workspace Launcher** dialogue box, during future start-ups this dialogue box will be by-passed and you will be taken directly to your workspace.
- You can have separate workspaces for separate projects. In this case, leave the **Workspace Launcher** dialogue box to appear on every start-up (so do not tick the above checkbox) and specify a new workspace when needed (the previously used workspace will be specified by default).

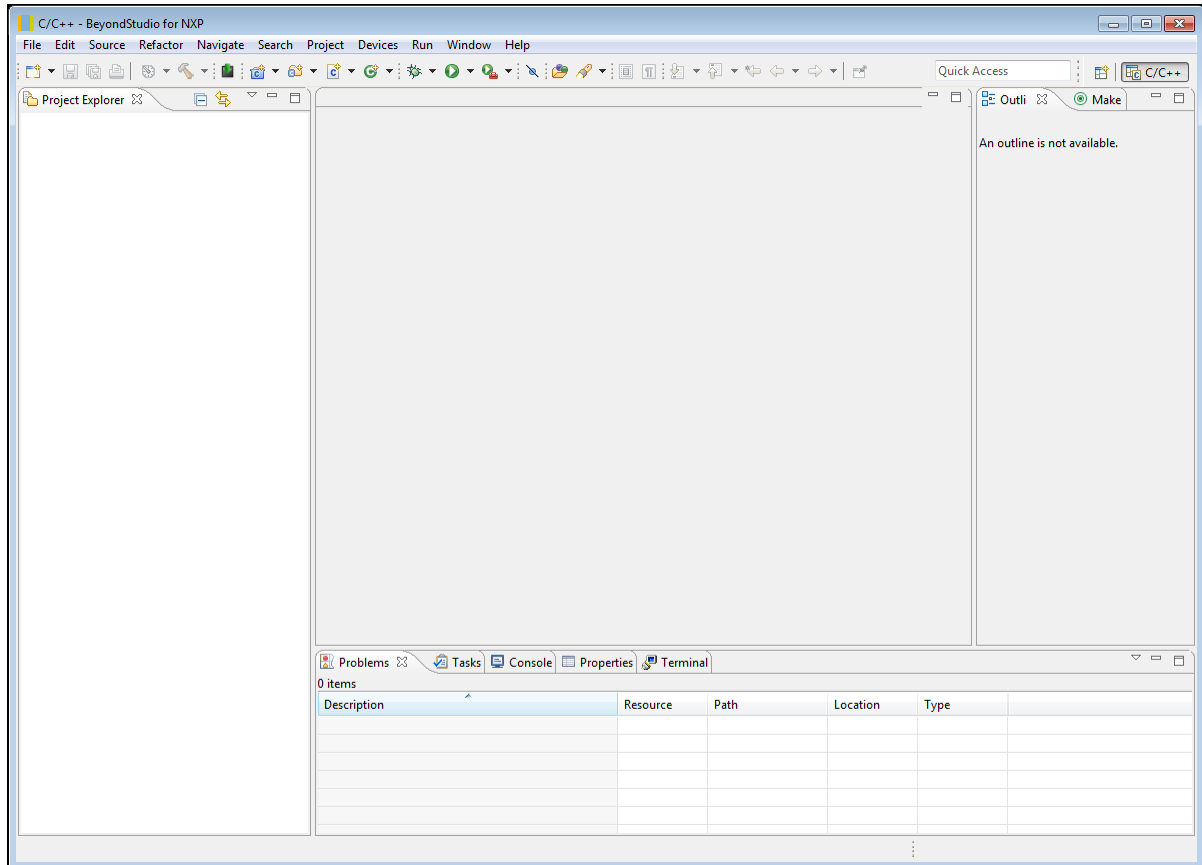


**Note 1:** BeyondStudio preferences can be easily shared by projects within the same workspace, allowing common settings between different projects (such as source formatting and debugger defaults).

**Note 2:** You can switch between workspaces using the menu option **File>Switch Workspace**. Even if the **Workspace Launcher** dialogue box is by-passed at start-up, you can also use this option to create a new workspace.

## 1.6 Workbench Perspectives and Layout

The main graphical interface of BeyondStudio is referred to as the 'workbench'. This interface can take one of a number of 'perspectives', where each perspective has a different use (e.g. debug). The most frequently used is the C/C++ perspective, shown below, which is used for general code development and editing.



**Figure 3: C/C++ Perspective of BeyondStudio Workbench**

The above figure shows the default layout of the C/C++ perspective. The panes or areas of the window are as follows:

- **Left (Project Explorer):** Provides a navigation tool through the workspace
- **Centre (Source):** Provides an editing space for the application source code and makefiles - multiple files can occupy separate tabs within this space
- **Right:** Contains multiple tabs of which Outline is the main one, allowing you to quickly navigate through the source files
- **Bottom:** Contains multiple tabs, including the **Console** tab, **Problems** tab and **Terminal** tab

The above panes can be rearranged, hidden and customised. Additional panes can also be selected and displayed by following the menu path **Window>Show view**.



**Tip:** Keyboard shortcuts can be used to streamline use of the workbench. The available shortcuts can be listed using the shortcut **Ctrl+Shift+L**. Shortcuts can also be customised and created by following the menu path **Window>Preferences** and then, within the **Preferences** dialogue box, **General>Keys**.



**Note:** The Debug perspective is also frequently used during application development. This perspective is illustrated and described in [Section 4.4](#).

## 2. Creating/Importing a Project

In BeyondStudio, an application under development is termed a project. Therefore, a project must first be created.

For JN516x devices, NXP provide Application Notes containing template and example applications that can be used as a starting point for custom application development. These applications are supplied with project files. You should therefore import a project from the relevant NXP Application Note rather than create a new project from scratch. These Application Notes are available from the NXP Wireless Connectivity TechZone and are listed below:

- JN-AN-1174: IEEE802.15.4 Application Template
- JN-AN-1190: JenNet-IP Application Template
- JN-AN-1135: Smart Energy HAN Solutions
- JN-AN-1189: ZigBee Home Automation Demonstration
- JN-AN-1171: ZigBee Light Link Solution
- JN-AN-1200: ZigBee RF4CE Application Template



**Note 1:** Not all of the above Application Notes may currently be compatible with BeyondStudio for NXP. Only those for the supported protocol stacks/profiles are compatible.

**Note 2:** If you have a JN516x project developed with the previous NXP SDK Toolchain (JN-SW-4041) that you wish to migrate to this toolchain, refer to the Application Note *BeyondStudio Migration Guidelines (JN-AN-1202)*.

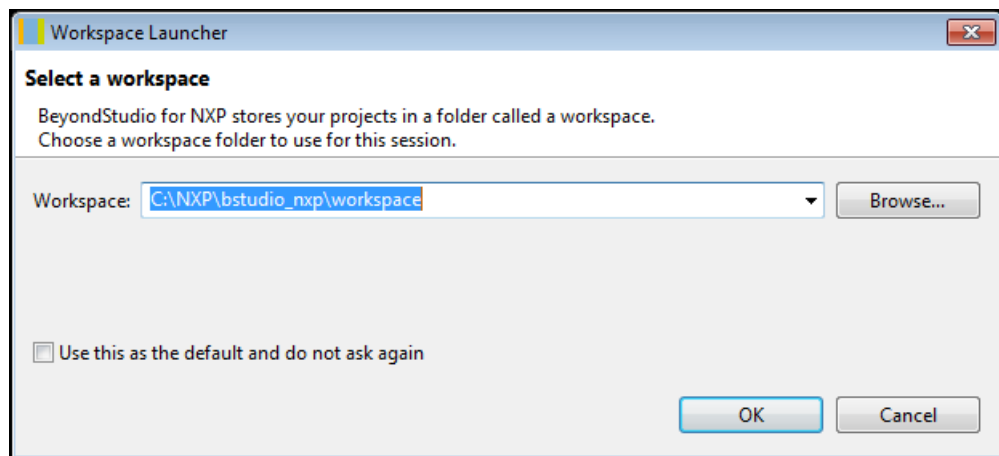
This chapter therefore describes how to start a new project by importing an existing project from an NXP Application Note.

## 2.1 Creating a Workspace

When starting your first project in BeyondStudio, you must create a workspace for the project. If this is not your first project, you may wish to use an existing work space or create a new workspace. For more information on workspaces, refer to [Section 1.5](#).

To create a workspace:

- Step 1** Launch BeyondStudio for NXP either via the Windows **Start** menu or by double-clicking on the desktop shortcut (if set up).
- Step 2** Select a workspace in the following **Workspace Launcher** dialogue box (which is displayed each time BeyondStudio is run, provided that it has not previously been disabled). The workspace normally sits in the BeyondStudio installation directory (e.g. under **C:\NXP\bstudio\_nxp**), as indicated in [Section 1.3](#), and can have any custom name. If you wish to locate your workspace elsewhere, you can specify another directory but you must later make certain modifications to the project, as described in [Appendix B](#).



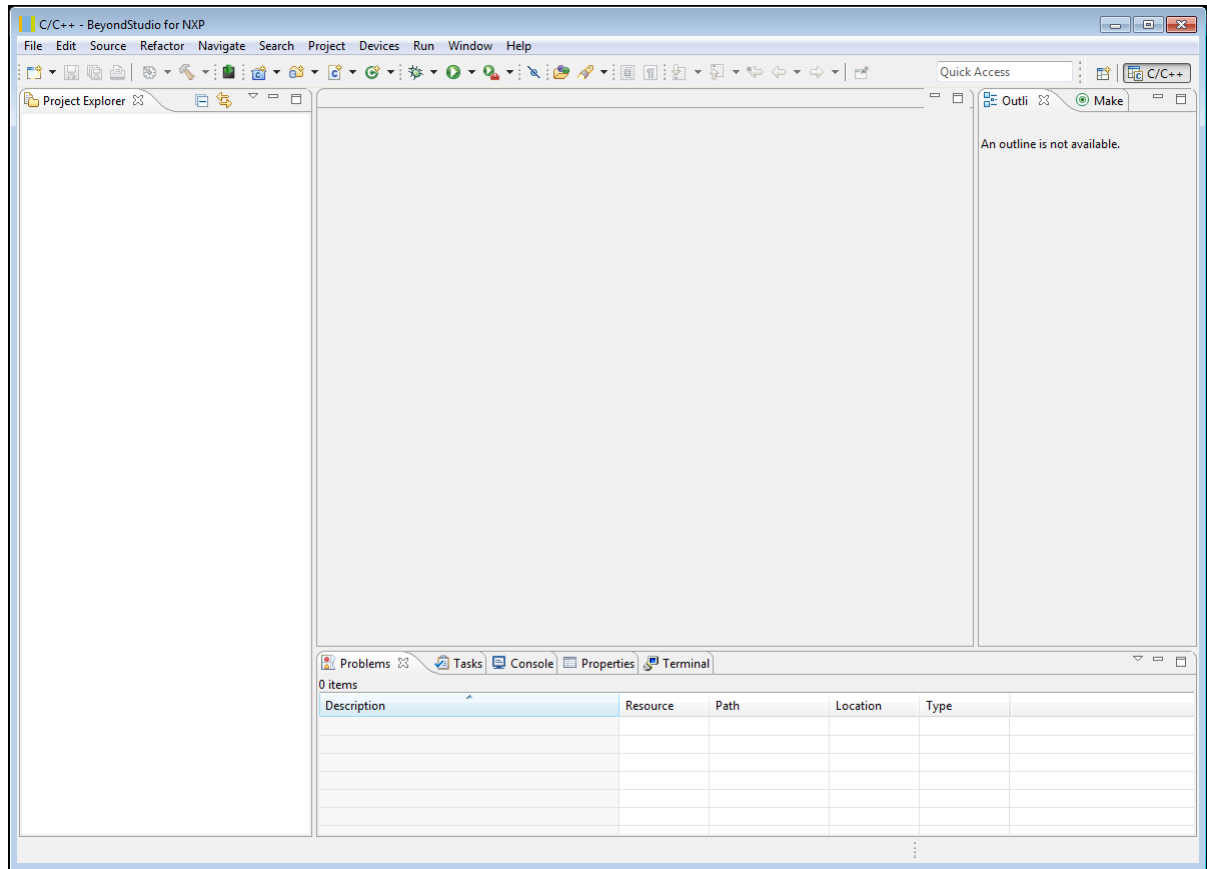
If you always want to keep your project files in this one workspace, tick the checkbox **Use this as the default and do not ask again** before clicking **OK** - this dialogue box will not then be displayed on subsequent launches of BeyondStudio.



**Note:** If the **Workspace Launcher** dialogue box does not appear and you wish to create a new workspace, you can do this via the menu option **File>Switch Workspace** once you are in the 'workbench' window (see next step).



**Step 3** The **Welcome** screen is next displayed. Close this window - this will take you to the 'workbench' window, which contains the default C/C++ perspective shown below.



**Note:** The C/C++ perspective is used for general code development. For information on workbench perspectives and layout, refer to [Section 1.6](#).

**Step 4** Configure the workspace preferences as follows:

- a) Open the **Preferences** dialogue box by following the menu path **Window>Preferences**.
- b) In the left tree of the **Preferences** dialogue box, open the **C/C++** entry and click on the **Indexer** sub-entry. The right side of the dialogue box is now populated with the Indexer options.
- c) In the section **Build configuration for the indexer**, select the radio-button **Use active build configuration**.
- d) Click **Apply** and then **OK**.

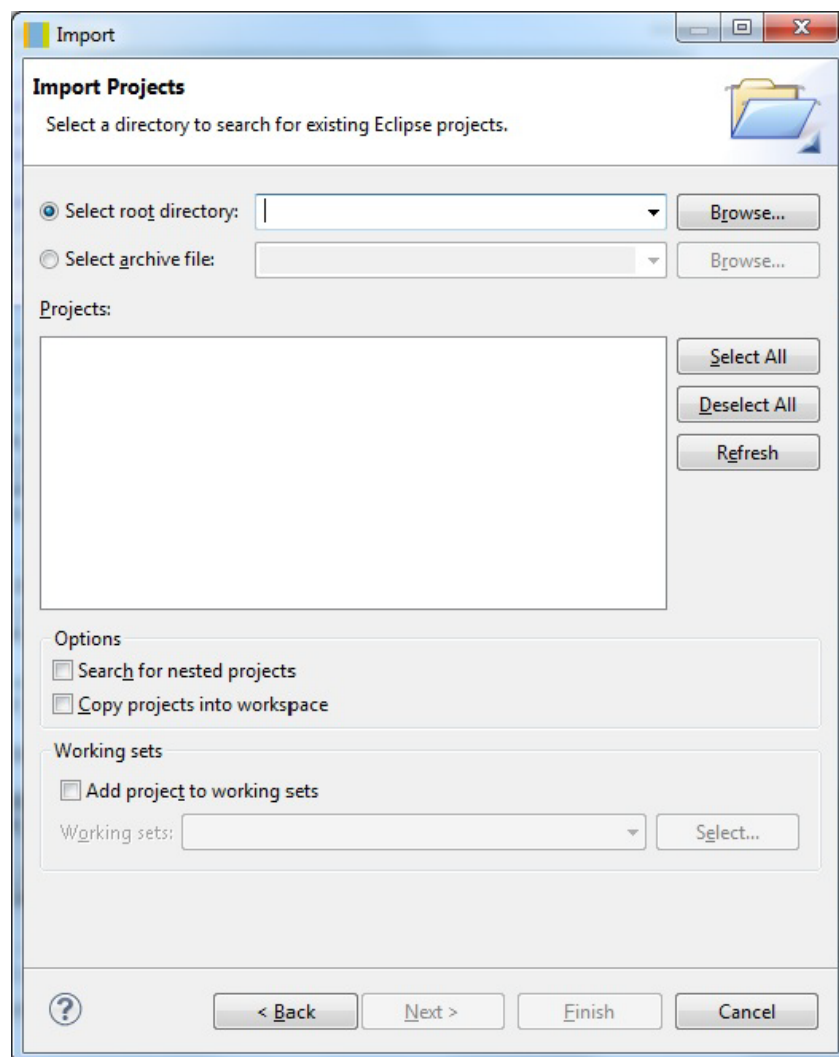
You can now use your workspace.

## 2.2 Importing a Project

You are advised to base your project on one provided in an existing NXP Application Note rather than create a project from scratch. This section describes how to import a project into BeyondStudio for NXP. You will need the Application Note ZIP file.

To import a project from an NXP Application Note:

- Step 1** Start BeyondStudio for NXP (if not already running) and follow the menu path **File>Import**. The **Import** dialogue box will be displayed.
- Step 2** Within the tree view in the **Import** dialogue box, follow the path **General>Existing Projects Into Workspace** and click **Next**. The **Import Projects** dialogue box will be displayed (shown below).



**Step 3** In the **Import Projects** dialogue box:

- a) Ensure that the **Select archive file** radio button is selected and browse to the ZIP file of the Application Note on which you are basing your project. The import wizard will then list the projects found in this Application Note in the **Projects** field of the dialogue box.
- b) In the **Projects** field, select the project to be imported and then click **Finish**.

**Step 4** Wait a moment while the project is imported. The project should appear in the left **Project Explorer** panel.

**Step 5** At this point, you should adapt the 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** pane 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 (contained in the relevant **Build** directory) and change the Target definition as illustrated below:

```
TARGET = myTargetName
```

- To add new source files (if any), follow the procedure in [Section 2.3](#). Then edit the associated makefile (contained in the relevant **Build** directory) and add the new source file to the Application Source section as illustrated below:

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



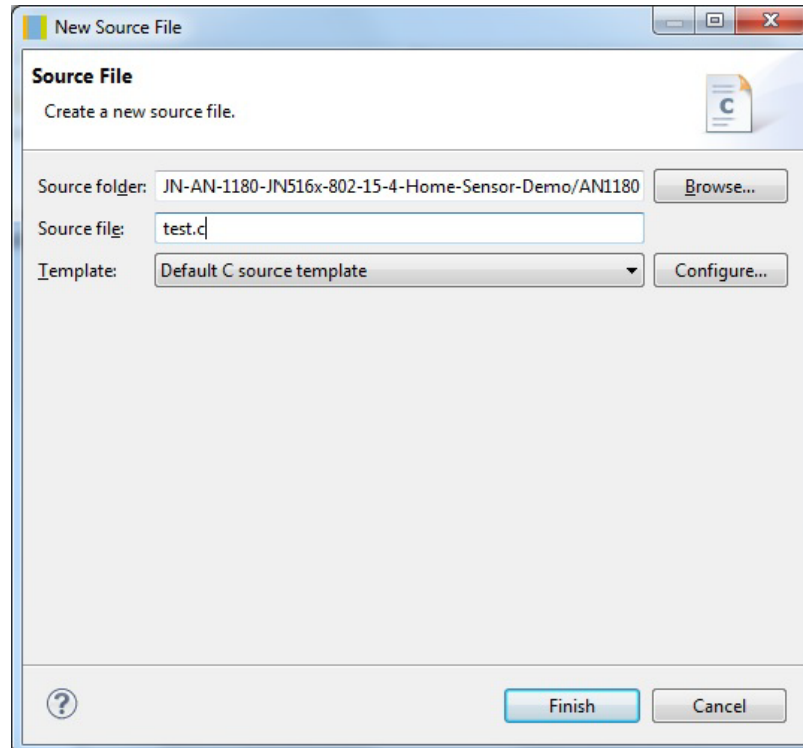
**Note 1:** To edit a file, such as a makefile, double-click on it in the **Projects Explorer** pane. The contents of the file will then appear in the central pane, where you can edit it - see [Section 3.1](#). Save your changes using the **Save** button on the toolbar.

**Note 2:** If the workspace for the project is located outside of the BeyondStudio installation directory, you will now need to make the modifications described in [Appendix B](#).

## 2.3 Adding a New Source File

To add a new C source file to a project:

- Step 1** In the **Projects Explorer** pane, expand the project folder so that the required **Source** sub-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**.



- 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** pane.
- 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 pane.
- Step 6** Edit the source file, as required.



**Note:** Before building the new source file, you will need to add it to the relevant makefile, as described in the final step in [Section 2.2](#).

## 3. Developing an Application

This chapter provides guidance on how to use BeyondStudio for NXP to develop an application project (that has been created as described in [Chapter 2](#)). The chapter describes:

- How to develop/edit application source code - see [Section 3.1](#)
- How to build the application (for release or debug purposes) - see [Section 3.2](#)
- How to load a built application into a JN516x device - see [Section 3.3](#)
- How to set up and execute a Run Configuration - see [Section 3.4](#)

### 3.1 Working on a Project

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

To edit your code, follow the procedure below:

- Step 1** If the required project is not already open, expand the project in the **Project Explorer** pane to display the project tree.
- Step 2** Double-click on the file to be edited in the **Project Explorer** pane. This displays a tab in the central pane.
- 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**.

- Step 4** You can now edit the code in the central pane. 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.


## 3.2 Building a Project

BeyondStudio for NXP allows a project to be built to produce a binary executable file in either of the following two build configurations:

- **Release configuration:** The application is built optimised for storage size and execution speed (as required for real-world deployment)
- **Debug configuration:** The application is built to incorporate information required for debugging and, as such, is not optimised for size or speed



**Caution:** Do not attempt to debug the Release build of an application, as the debugger will not operate with a Release build.

The **Build** button ('hammer'  icon) on the toolbar is used to initiate a build.

To build your project:

**Step 1** Select the required build type (Release or Debug) as follows:

- For Debug, ensure that the relevant makefile(s) contain the following line (if this line is present but commented out, uncomment it):

```
DEBUG ?= HW
```

Also ensure that the DEBUG variable is not over-ridden by the project configuration. Follow the menu path **Project>Properties** to display the **Properties** dialogue box and in the left tree follow **C/C++ Build>Environment**. Check that the DEBUG variable is not in the **Environment variables to set** list and if it is, highlight it and click on **Undefine**, then **Apply**, then **OK**.

The line `DEBUG_PORT ?= UART0` must also be present in the makefile.

- For Release, ensure that the `DEBUG ?= HW` line is not present or is commented out in the relevant makefile(s)

**Step 2** Start the build by either of the following methods:

- Click the the **Build** button on the toolbar and select the project to be built - the application will then build automatically.
- In the **Projects Explorer** pane, right-click on the relevant project and, from the pop-up menu, select **Build Project** - the application will then build automatically.

**Step 3** Check the progress of the build in the bottom pane of the workbench. Standard output from the build can be seen on the **Console** tab. Any errors/warnings created by the build process will be displayed on the **Problems** tab.

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



**Note:** Following one or more builds with the Debug configuration, the files created during the debug process can be deleted via the menu path **Project>Clean**. You should do this before rebuilding with the Release configuration to produce the final output.

---

### 3.3 Loading an Application into a Device

A built application can be loaded into the internal Flash memory of a JN516x device directly from BeyondStudio for NXP using its integrated Flash programming tool. This tool is useful during application development and testing.



**Note 1:** The JN51xx Flash programmer incorporated in BeyondStudio for NXP can be used to program JN516x internal Flash memory only, and not external Flash memory. To program an external Flash device attached to the JN516x microcontroller, the JN51xx Production Flash Programmer should be used (see Note 2 below).

**Note 2:** A CLI Flash programming tool is available for production situations. This software has the part number JN-SW-4107 and is described in the *JN51xx Production Flash Programmer User Guide (JN-UG-3099)*, which are both available from the NXP Wireless Connectivity TechZone.

The integrated Flash programming tool also offers the facility to erase or program the JN516x EEPROM and to program a customer IEEE/MAC address into the target device. These items can be programmed at the same time as programming Flash memory or without programming Flash memory.


Before loading a built application, ensure that you have the following:

- A target device containing a JN516x microcontroller
- A suitable cable for connecting a USB port of your PC and to the target device (a cable is not needed when programming a JN516x USB dongle)
- The FTDI USB-to-serial device driver on your PC (guidance on FTDI device driver installation is given at the end of [Section 1.2.1](#))
- The built application **.bin** file to be loaded (this file is located in the **Build** directory for the project)
- If applicable, the file containing data to be loaded into EEPROM and/or the customer IEEE/MAC address to be programmed

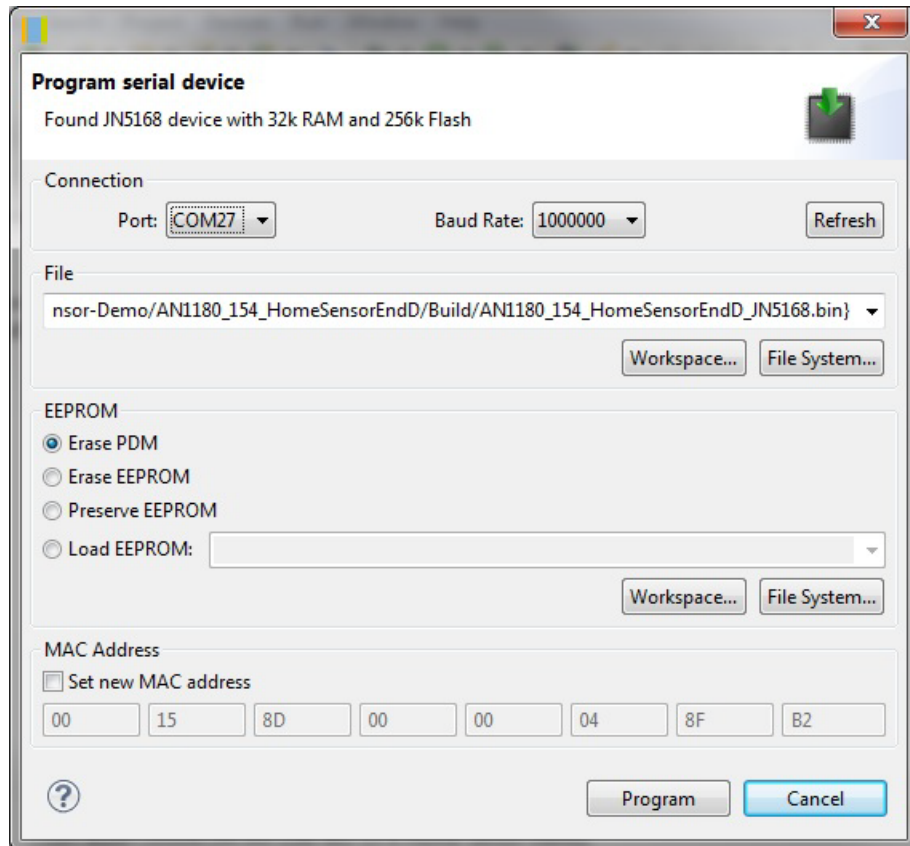
### Chapter 3

#### Developing an Application

To load a binary application into the Flash memory of the target JN516x device:

- Step 1** Connect the target JN516x device (which may be mounted on a board or module) to a USB port of your PC (UART0 on the JN516x device is used for this connection). At this point, you may be prompted to install the driver for the cable (if used).
- Step 2** Determine which serial communications port on your PC has been allocated to the USB connection - to identify the relevant port, refer to [Appendix A](#).
- Step 3** If not already running, launch BeyondStudio for NXP.
- Step 4** Start the Flash programming tool within BeyondStudio for NXP by any one of the following methods:
- Follow the menu path **Devices>Program Device**
  - Click on the **Program Device** button (  icon) in the toolbar
  - Use the keyboard shortcut **Ctrl-6**

The **Program serial device** dialogue box will now appear (shown below).





**Step 5** In the **Program serial device** dialogue box (shown above), configure the Flash programming tool as follows:

- **Connection** - Fill in the **Port** and **Baud Rate** fields as follows:
  - **Port** - Select the serial communications port used for the connection, as determined in Step 2
  - **Baud Rate** - Select the required data rate, in bps (normally, the default value of 1000000 should be used)
- **File** - Navigate to the binary file to be loaded (if this file is in your workspace, use the **Workspace** button, otherwise use the **File System** button)
- **EEPROM** - Select the required operation on JN516x EEPROM:
  - **Erase PDM** to delete all persistent data records from EEPROM
  - **Erase EEPROM** to delete all the contents of EEPROM
  - **Preserve EEPROM** to keep all the contents of EEPROM
  - **Load EEPROM** to load/replace data in EEPROM (in this case, navigate to the data file to be loaded - if this file is in your workspace, use the **Workspace** button, otherwise use the **File System** button)
- **MAC Address** - The IEEE/MAC address of the target device will be displayed. If you wish to over-ride the factory-set MAC address with a customer MAC address, tick the **Set new MAC address** check-box and enter the new MAC address into the eight boxes (note that this is only one-time programmable).

**Step 6** Click the **Program** button. The dialogue box will now disappear and loading will start. The progress of loading will be displayed in a progress box.

**Step 7** Once the loading has successfully completed, disconnect the serial cable and, if required, reset the target device to run the application.



**Caution:** *If set, a customer MAC address will over-ride but not over-write the factory-set MAC address. Both MAC addresses will still be held in the device but the customer MAC address will be used. Once written, the customer MAC address cannot be changed or erased.*



**Note 1:** Optionally, a **Terminal** tab in the workbench can be associated with a connected device, in order to allow output from an application running on the device to be displayed. This option can be configured as described in [Appendix D.1](#).

**Note 2:** BeyondStudio for NXP also allows you to read the contents of Flash memory and/or EEPROM of a JN516x device and output this data to the file system of your PC. This type of memory access is described in [Appendix E](#). The tool can also be used to later write this data back to EEPROM, to return to a known state.

## 3.4 Creating and Using a Run Configuration

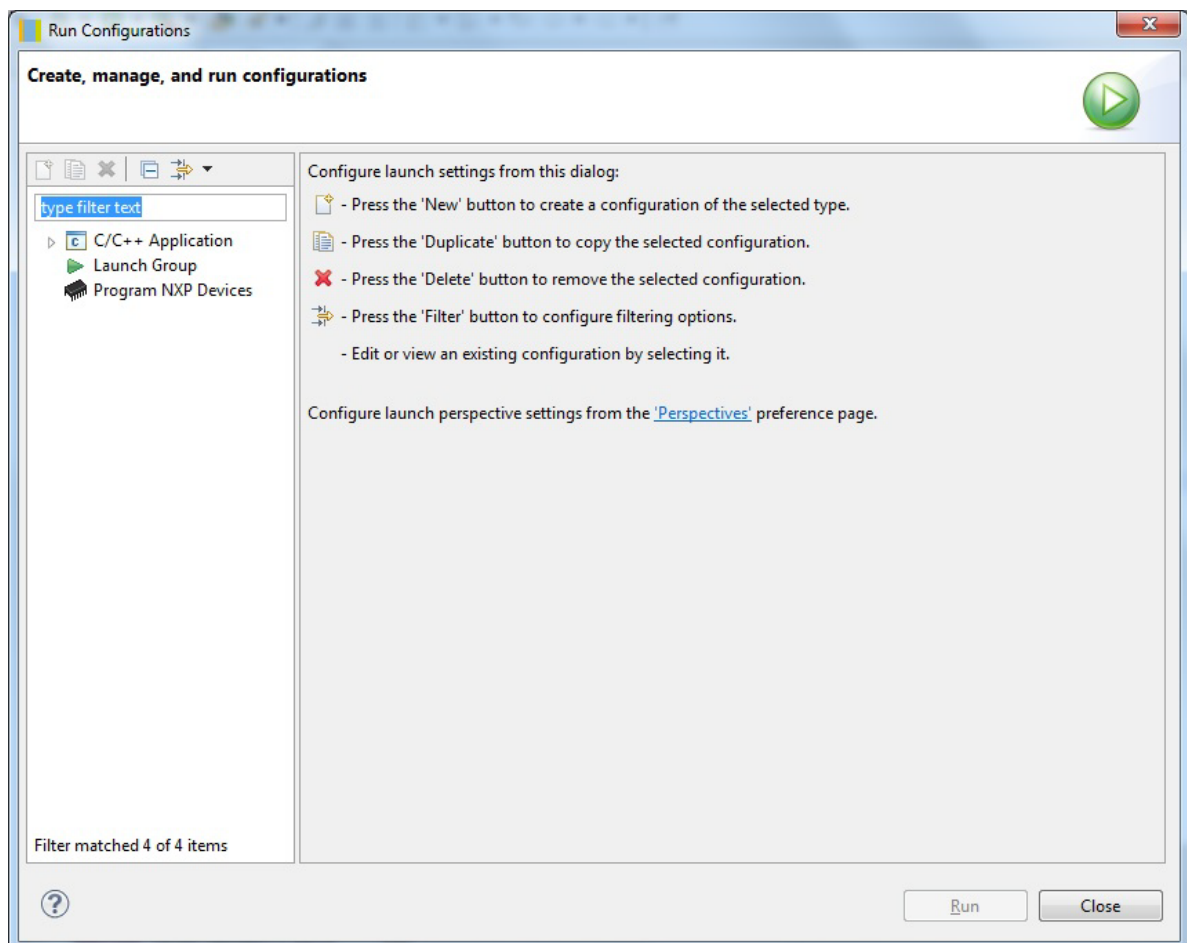
BeyondStudio for NXP allows you to create a 'Run Configuration' for one or more applications. A Run Configuration allows an application to be built, loaded into the target device and run as the result of a single click of a button in the BeyondStudio workbench. If the Run Configuration includes multiple applications and devices, all the applications will be built and loaded into their respective devices in parallel.

Note that in order to create and use a Run Configuration:


- The device driver for an FTDI USB-to-serial chip must be present on your PC. Guidance on FTDI device driver installation is given at the end of [Section 1.2.1](#).
- The target devices must be connected to the PC and the serial communications ports to which the devices are connected must be determined in advance, as described in [Appendix A](#).

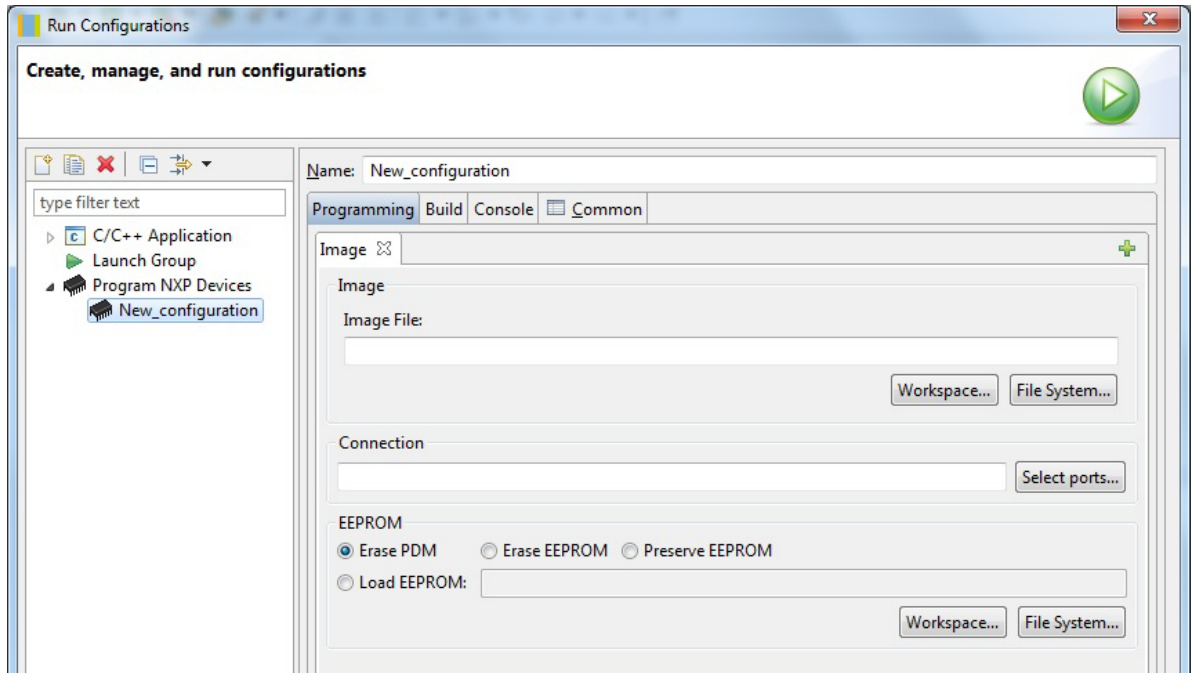
To set up and execute a Run Configuration:

**Step 1** Within BeyondStudio for NXP, follow the menu path **Run>Run Configurations**. This will launch the **Run Configurations** dialogue box (shown below).



**Step 2** In the **Run Configurations** dialogue box (shown above), create a new Run Configuration as follow:

- a) In the left pane of the dialogue box, click on **Program NXP Devices**.
- b) Click on the **New** button (  icon) at the top of the left pane of the dialogue box. The right side of the dialogue box will now be populated with fields and tabs (as shown below).



**Step 3** In the **Name** field, replace 'New\_configuration' with the name of your Run Configuration.

**Step 4** On the **Programming** tab, fill in the required information (for one application and target device) in the fields on the **Image** sub-tab, as follows:

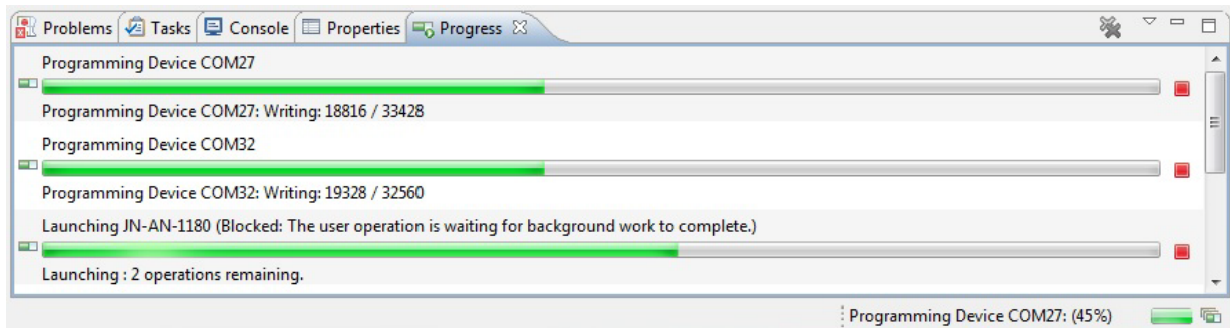
- **Image** - Navigate to the binary file to be loaded (if this file is in your workspace, use the **Workspace** button, otherwise use the **File System** button)
- **Connection** - Select the serial communications port(s) used for the connection to the target device(s). Multiple ports can be selected using the **Select ports** button and appear in the field as a comma-separated list
- **EEPROM** - Select the required operation on JN516x EEPROM:
  - **Erase PDM** to delete all persistent data records from EEPROM
  - **Erase EEPROM** to delete all the contents of EEPROM
  - **Preserve EEPROM** to keep all the contents of EEPROM
  - **Load EEPROM** to load/replace data in EEPROM (in this case, navigate to the data file to be loaded - if this file is in your workspace, use the **Workspace** button, otherwise use the **File System** button)

Then click the **Apply** button.

### Chapter 3

#### Developing an Application

- Step 5** If you wish to include another application and/or device in the Run Configuration, click the **+** symbol on the right to add another **Image** sub-tab and repeat Step 4.
- Step 6** Finally, to execute the Run Configuration, click the **Run** button, otherwise click the **Close** button. The dialogue box will disappear. If run, the progress of execution for all the applications/devices will be displayed in the **Progress** tab in the bottom pane of the workbench (see example below). The **Progress** tab is not visible by default but can be displayed by following the menu path **Window>Show View>Other** and then in the **Show View** dialogue box selecting **General>Progress**.



**Note 1:** A Run Configuration can be deleted by first highlighting it in the tree view in the left pane and then either clicking the **Delete** button ( **X** icon), or by right-clicking and selecting Delete from the pop-up menu.

**Note 2:** Optionally, a Run Configuration can associate a 'Terminal' tab in the workbench with a connected device, in order to allow output from an application running on the device to be displayed. This option can be configured as described in [Appendix D.2](#)

## 4. Debugging an Application

This chapter describes how to use the debugging features of BeyondStudio for NXP. This allows an application which has been built with the Debug option enabled (see [Section 3.2](#)) to be run on a connected JN516x device and debugged through interactions via the Debug perspective of BeyondStudio.



**Note:** Debugging of application code running on a JN516x device is conducted via JTAG hardware. Basic information is provided here but for full details, refer to the Application Note *JN516x JTAG Debugging in BeyondStudio (JN-AN-1203)*.


### 4.1 Creating a Debug Configuration

In order to debug an application, you must first set up a Debug Configuration for the built application. Debug Configuration defaults are available within BeyondStudio. Therefore, there is no need to create a Debug Configuration from scratch but you still need to add a Debug Configuration for your application.



**Note:** You will create a Debug Configuration for each application binary within your project. Therefore, the application(s) must be built for Debug before creating the Debug Configuration(s).

To add a Debug Configuration for your application:

- Step 1** Make sure your project is open in BeyondStudio for NXP and, if not already done, build the application(s) for debug, as described in [Section 3.2](#).
- Step 2** Highlight your project in the **Project Explorer** pane, right-click and select **Refresh** - the binaries should now appear in the project tree (this may happen automatically).
- Step 3** Click the down-arrow to the right of **Debug** button ('bug'  icon) on the toolbar in order to access the drop-down menu. In this menu, select **Debug As** - this will display a list from which you must select **NXP JN516x Application**. The **Beyond BA Application** dialogue box will now be displayed.
- Step 4** In the **Beyond BA Application** dialogue box, a list of **.elf** files will be displayed corresponding to the built applications that BeyondStudio was able to find in the project area. Select the file corresponding to the application for which you wish to create a Debug Configuration and click **OK**.
- Step 5** If required, repeat Steps 3 and 4 for any other built application that needs a Debug Configuration.

## 4.2 Connecting PC to JN516x Device for Debugging

The Debug build of the application will be run on a JN516x device connected to the PC. The Debug binary file can be loaded into the JN516x device via a normal USB connection but the connection for debugging purposes must be made via JTAG hardware (see Note 4 below). This section outlines the connection procedure, but a more detailed account is provided in the Application Note *JN516x JTAG Debugging in BeyondStudio* (JN-AN-1203).



**Note 1:** To debug applications running on boards of the NXP JN516x-EK001 Evaluation Kit, you will also need a DR1222 JTAG Expansion Board and a JTAG adaptor. The recommended adaptors are the Beyond Debug Key and Amontec JTAGkey/JTAGkey2.

**Note 2:** During debug, the JN516x UART0 is not available to the application for serial output. Instead, the application should use either UART1 (e.g. routed to the USB connector on a DR1199 Generic Expansion Board) or the **DBG\_Printf()** function which redirects output via the JTAG interface. If UART1 is used, the serial output is received by the PC via a USB port and can be displayed in BeyondStudio via a Terminal tab associated with the relevant serial communications port, as described in [Appendix D](#).

**Note 3:** Only one Debug Configuration (for one application) can be active in each instance of BeyondStudio. Debugging applications on multiple devices simultaneously is not recommended, as setting up the JTAG hardware becomes complicated.

**Note 4:** The Debug application **.bin** file, produced as described in [Section 3.2](#), contains only instructions to enable the JTAG interface, and no application code. This file must be loaded into the JN516x device using a normal serial connection and only needs to be loaded once per device. The Debug application image (which was also produced in the build process) is subsequently loaded via the JTAG interface on starting debug.

- Step 1** Load the Debug binary file into the JN516x device via a serial connection using the Flash programming tool within BeyondStudio for NXP, as described in [Section 3.3](#). Refer to Note 4 above for information on the purpose of this file.
- Step 2** Once the binary file has been loaded, remove the USB cable from the target device and if any other power source is present for the device, remove it (e.g. remove any batteries). **There must be no power to the device during the next step.**



- Step 3** Enable the target device for JTAG debugging. In the case of a JN516x module that is fitted to a DR1174 Carrier Board from an NXP JN516x-EK001 Evaluation Kit:
- Put the JTAG jumper on the carrier board into the EN (enable) position, as shown in the photograph below.



- Install a DR1222 JTAG Expansion Board onto the carrier board (this expansion board uses the JTAG connector of the carrier board).
- You can optionally install further expansion boards on top of the JTAG Expansion Board, depending on the needs of your application.
- Connect the PC to a suitable JTAG adaptor/dongle (such as the Beyond Debug Key or the Amontec JTAGkey or JTAGkey2) via a USB cable.



**Note:** The first time that you connect the JTAG adaptor to your PC, the relevant USB driver must be installed. This installation can be performed using the Zadig tool provided with BeyondStudio for NXP. Run Zadig and select the connected adaptor (you may need to select **Options>List All Devices** to include the adaptor in the list), then change the selected driver to **libusb-win32** and click **Replace Driver**. For full details of this installation, refer to the Application Note JN-AN-1203.

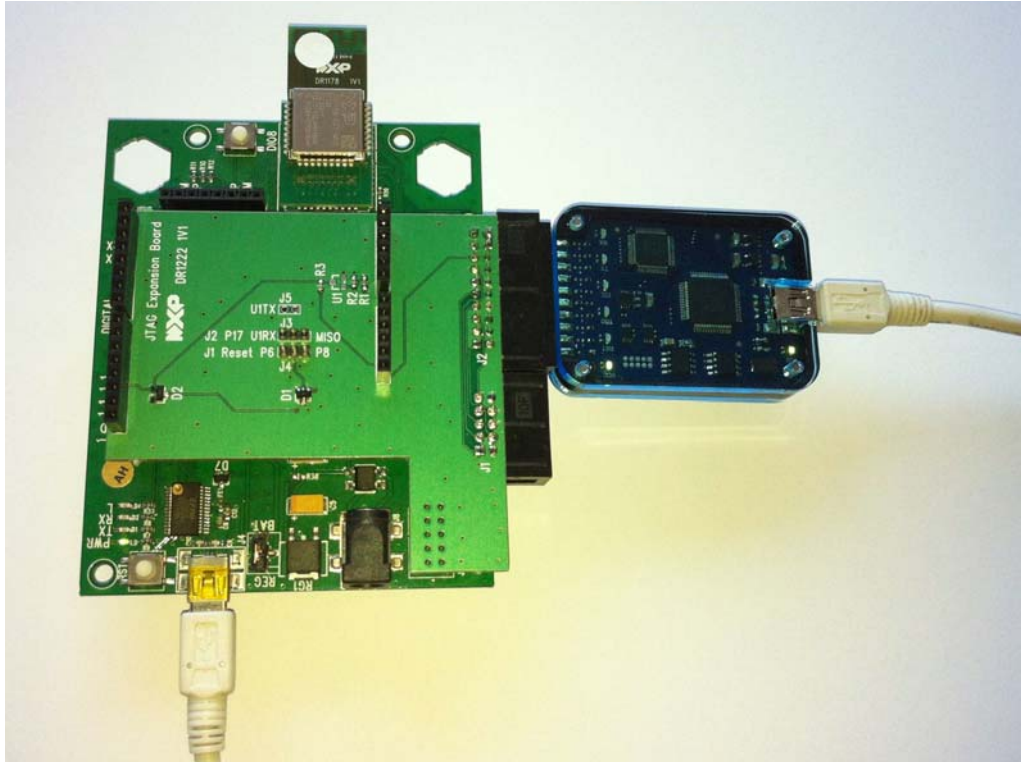
- Connect the JTAG adaptor to the carrier board's 20-way JTAG header (thus, the PC is now connected to the board via a USB cable and the JTAG adaptor).

- Step 4** If the target device is self-powered (e.g. from batteries), reconnect this power source. Otherwise, reinstate the USB connection (disconnected in Step 2) to the PC in order to supply power to the device. If the USB connection is reinstated, it is only used to power the device during debug and cannot be used for serial output from the application (see Note 2 on page 38).

## Chapter 4

### Debugging an Application

The photograph below shows the final connections (in this case, with the board powered from a USB port of the PC).



**Note:** The JN516x bootloader now waits for BeyondStudio to connect via the JTAG interface. This occurs when the debugger is launched in BeyondStudio, as described in [Section 4.3](#).



## 4.3 Launching the Debugger


Once a Debug Configuration has been created for a built application (see [Section 4.1](#)) and the JTAG hardware has been set up and enabled by loading the Debug binary file into the J516x device (see [Section 4.2](#)), the debugger can be started within BeyondStudio for NXP.



**Note:** On launching the debugger, the application image to be debugged will be automatically loaded into the JN516x device via the JTAG interface and run (but will be paused at the **AppColdStart()** function).

To launch the debugger within BeyondStudio for NXP:

**Step 1** Make sure your project is open in the C/C++ perspective of BeyondStudio for NXP.

**Step 2** Click the down-arrow to the right of **Debug** button ('bug'  icon) on the toolbar in order to access the drop-down menu. The applications with Debug Configurations will be listed at the top of this menu - select the required application from this list.




**Note:** If you simply click on the **Debug** button (and not the arrow), the last application that was debugged will be automatically selected.

**Step 3** BeyondStudio will now try to switch to the Debug perspective but will request permission by displaying the **Confirm Perspective Switch** dialogue box. Accept this switch by clicking the **Yes** button (you may also wish to tick the **Remember my decision** checkbox so that this dialogue box is not displayed when the debugger is launched in the future).

BeyondStudio for NXP will now display the Debug perspective, as illustrated and described in [Section 4.4](#).

**Step 4** The Debug application image will be automatically loaded into the JN516x device and run. Initially, application execution will be paused at the **AppColdStart()** function and will wait for user interaction. The application can now be debugged via the Debug perspective.



**Important:** When a Debug session is no longer required, it must be ended using the **Terminate** button ( icon) on the toolbar. You can then revert back to the C/C++ perspective - for example, by following the menu path **Window>Open Perspective>C/C++**. You should load a Release build of the application into the JN516x device in order to deactivate the JTAG hardware.

## 4.4 Debug Perspective

Once the debugger has been launched in Beyond Studio for NXP (as described in [Section 4.3](#)), the workbench will take the form of the Debug perspective, shown below.

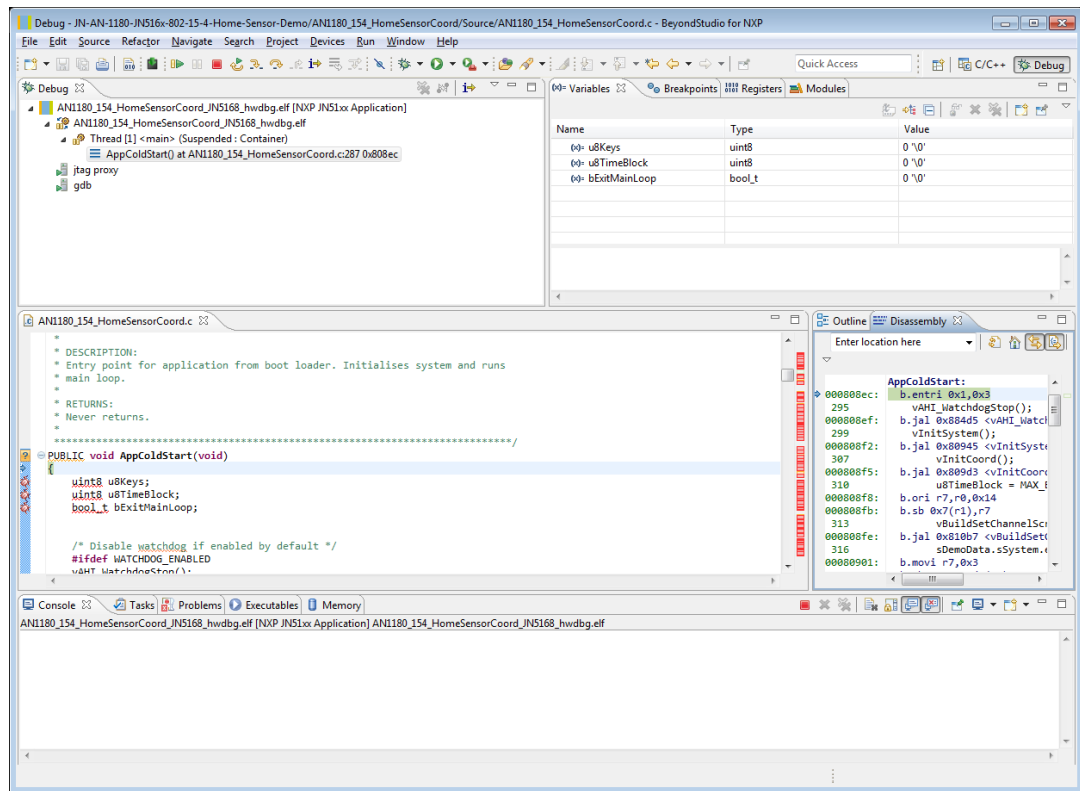


Figure 1: Debug Perspective of BeyondStudio Workbench

The panes or areas of the above window are as follows:

- **Top-left (Debug):** Displays the processor stack and allows selection of any stack frame (which will update the other displays to the selected frame)
- **Top-right:** Contains a number of tabs, including:
  - **Variables:** Shows local variables within the selected stack frame
  - **Breakpoints:** Shows all currently configured breakpoints
  - **Registers:** Displays the contents of the CPU registers
- **Middle-left (Source):** Displays the C source code at the location where the processing is stopped
- **Middle-right:** Contains a number of tabs including the **Disassembly** tab which displays the assembly language at the location where the CPU is paused
- **Bottom:** Contains multiple tabs, including the **Console** tab, **Problems** tab and **Terminal** tab

During debug, output from the JenOS Debug API function **DBG\_vPrintf()** is directed to the **Console** tab.

## 4.5 Debug Operation Summary

The table below lists and outlines the operations that are commonly used during debug. For more detailed information on these operations, refer to the Application Note *JN516x JTAG Debugging in BeyondStudio (JN-AN-1203)*.



**Important:** The JN516x CPU is automatically stalled at the **AppColdStart()** function and should be started manually. If the CPU is subsequently stalled (e.g. when paused or at a breakpoint), JN516x peripherals (such as timers) continue to run. Also, if the device is a network member, the network will perceive the device to be lost.

Operation	Icon	Description
Run		The CPU free-runs through the application code. This processing will continue until a breakpoint is reached or a Pause operation is performed.
Pause		The CPU pauses free-run processing of the application code. Note that resuming processing may cause problems (see important note above) and it is advisable to reset the device after a pause.
Reset		Resets the device and performs a cold start.
Step into		Executes the next instruction in the application code and if this is a subroutine, steps into the routine (executes its first instruction and stops).
Step over		Executes the next instruction in the application code and if this is a subroutine, steps over the routine (executes it in its entirety without stopping).
Set breakpoint	-	To set a breakpoint, locate the relevant line in the source pane (or Disassembly tab) and double-click in the margin - a blue circle appears to indicate the presence of a breakpoint. The breakpoint is also listed in the Breakpoints tab. A maximum of 4 breakpoints are supported by the JN516x device (any additional breakpoints will be ignored).
View local variables	-	Local variables contained in a stack frame created by the latest function call can be viewed in the Variables tab.
View global variables	-	Global variables can be viewed in the Expression tab.

**Table 1: Debug Operation Summary**

In addition to the above operations, the following debug features are available:

- **Catch Exceptions:** An exception will cause the CPU to stall - this will be indicated in the stack information in the Debug pane.
- **Enable Watchdog:** The JN516x Watchdog timer is normally enabled by default but is stopped by the bootloader for a Debug build of an application. If this timer is required during debug, it must be re-enabled in the application using the function **vAHI\_WatchdogStart()**. It must also be configured using the function **vAHI\_WatchdogException()** to cause an exception rather than a reset. Both functions are provided in the JN516x Integrated Peripherals API.



---

## Appendices

---

### A. Identifying the PC Communications Port Used

When connecting a USB port of your PC to a serial device, you need to find out which serial communications port your PC has allocated to the connection, as described below (for Windows 7 and 8).

- Step 1** In the Windows **Start** menu, open the **Control Panel** by following the menu path:  
**Start>Control Panel** (Windows 7) or  
**Start>All Apps>Control Panel** (Windows 8)
- Step 2** From the **Control Panel**, open the **Device Manager** by following the path:  
**System>Device Manager** (Windows 7) or  
**System & Security>Device Manager** (Windows 8)
- Step 3** Within 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 serial device (it will be labelled 'USB Serial Port') and make a note of it (e.g. COM1).

---

### B. Using a Workspace Outside the Installation Folder

The workspace for a project normally sits in the BeyondStudio installation directory (e.g. under **C:\NXP\bstudio\_nxp**) but can be located elsewhere. If you wish your workspace to be located outside the installation directory, proceed as follows:

- Step 1** Create your workspace as described in [Section 2.1](#), specifying your chosen directory outside of the BeyondStudio installation directory.
- Step 2** Create/import your project in BeyondStudio as described in [Section 2.2](#).
- Step 3** Follow the menu path **Project>Properties** (or right-click on your project in the **Projects Explorer** pane and select **Properties** from the pop-up menu). The **Properties** dialogue box for your project will now appear.
- Step 4** In the left pane of the **Properties** dialogue box, open **NXP JN516x Applications** in the tree and, underneath, click on **Target Stack**. The **Target Stack** section will now be displayed on the right side of the dialogue box.
- Step 5** In the **Target Stack** section, tick the **Set stack environment variables** checkbox. This section also allows you to select the NXP SDK with which your project will be built. Then click **Apply** followed by **OK**.

Ticking the checkbox over-rides the **SDK\_BASE\_DIR** variable in the project makefiles. The stack selection allows the **JENNIC\_SDK** variables in the project makefiles to be over-ridden.

---

## C. Eclipse Enhancements

BeyondStudio for NXP is based on the Kepler release of the Eclipse IDE and provides enhancements over the version of Eclipse supplied in the previous NXP SDK Toolchain (JN-SW-4041). This appendix describes these enhancements:

- Serial Terminal
- Run Configurations
- Flash programmer
- Flash Memory and EEPROM Dump

---

### C.1 Integrated Serial Terminal

BeyondStudio for NXP includes a serial terminal which can be used to capture serial output from connected devices, and to send input to them. The terminal emulator is VT100 compatible. Setting up a serial terminal is described in [Appendix D](#).

---

### C.2 Run Configurations

BeyondStudio for NXP allows you to create a 'Run Configuration' for one or more applications. This allows an application to be built, loaded into the target device and run as the result of a single click of a button. If a Run Configuration includes multiple applications and devices, all the applications will be built and loaded into their respective devices simultaneously. Creating and using a Run Configuration are described in [Section 3.4](#).

---

### C.3 Flash Programmer

BeyondStudio for NXP includes a Flash programming tool which is integrated into the graphical interface of the IDE. This can be used to load a built application into the internal Flash memory of a JN516x device, but also has functionality to manipulate JN516x EEPROM, as follows:

- Erase PDM records: Delete all persistent data records from EEPROM
- Erase EEPROM: Delete all the contents of EEPROM
- Preserve EEPROM: Keep all the contents of EEPROM
- Load EEPROM: Load/replace data in EEPROM

Use of the integrated Flash programmer is fully described in [Section 3.3](#).

---

### C.4 Flash Memory and EEPROM Dump

BeyondStudio for NXP allows you to read the contents of the Flash memory and/or EEPROM of a JN516x device and dump the output to the file system on your PC. Use of this feature is described in [Appendix E](#).

## D. Creating and Using Serial Terminals

BeyondStudio for NXP can provide a serial terminal emulator which allows the display of output from an application running on a connected device, as well as input to the application typed into the PC keyboard. This terminal emulator appears as a 'Terminal' tab in the bottom pane of the workbench. It is possible to have multiple Terminal tabs associated with multiple devices. Each Terminal tab is associated with the serial communications port to which a device is connected - the relevant port number is displayed at the top of the Terminal tab 'screen'.

The sub-sections below describe how to create/enable a Terminal tab to receive the output from an application running on a connected device.




**Note 1:** The following procedures assume that the devices have been connected to the PC and the serial communications ports to which the devices are connected have been determined as described in [Appendix A](#).

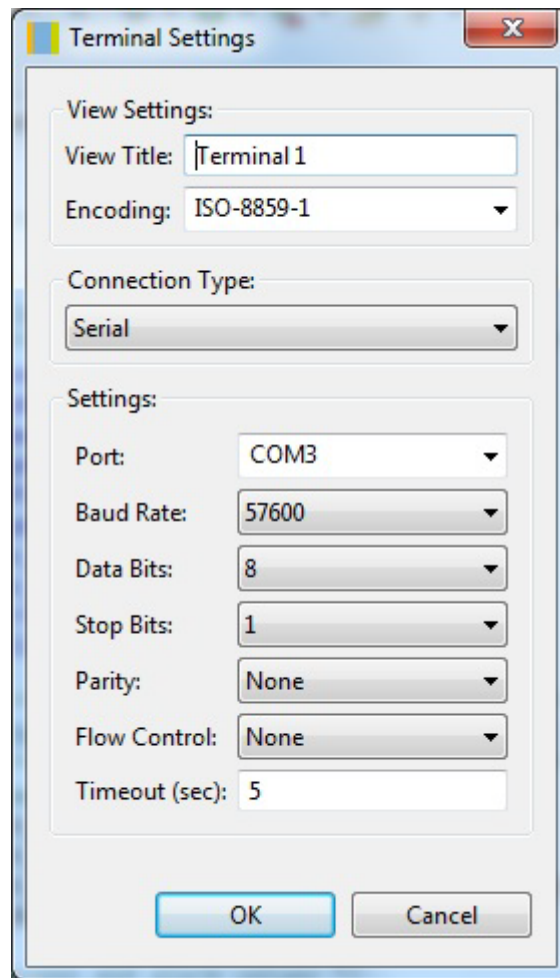
**Note 2:** A Terminal tab will be automatically disconnected from the associated serial port when the integrated Flash programming tool is used. The connection will be automatically re-established when the Flash programming has completed.

### D.1 Creating/Enabling a Terminal Tab

A Terminal tab is provided by default in the bottom pane of the workbench - this first such tab is numbered **Terminal 1**. To use this tab, it must be associated with a serial port/device. Further Terminal tabs can also be created for other serial connections.

To enable or create a Terminal tab and associate it with a particular serial port/device:

- Step 1** To enable the default Terminal tab in BeyondStudio for NXP, click on it or follow the menu path **Window>Show View>Terminal**.
- Step 2** To associate this Terminal tab with a device, click the **Settings** button (  icon) on the toolbar in the top-right of the pane. This displays the **Terminal Settings** dialogue box (shown below).



**Step 3** In the **Terminal Settings** dialogue box (shown above), provide the following information:

- a)** In the **Port** field, from the drop-down list, select the serial communications port used for the connection to the device to be associated with the Terminal tab.
- b)** In the **Baud Rate** field, from the drop-down list, select the baud rate for the connection between the device and Terminal tab.

If you wish, you may also change the name of the tab in the **View Title** field.

Click **OK** to save the settings.

**Step 4** Connect the terminal to the associated serial communications port by clicking the **Connect** button (🔌 icon) in the top-right of the pane (also see Note 2 on page 47).

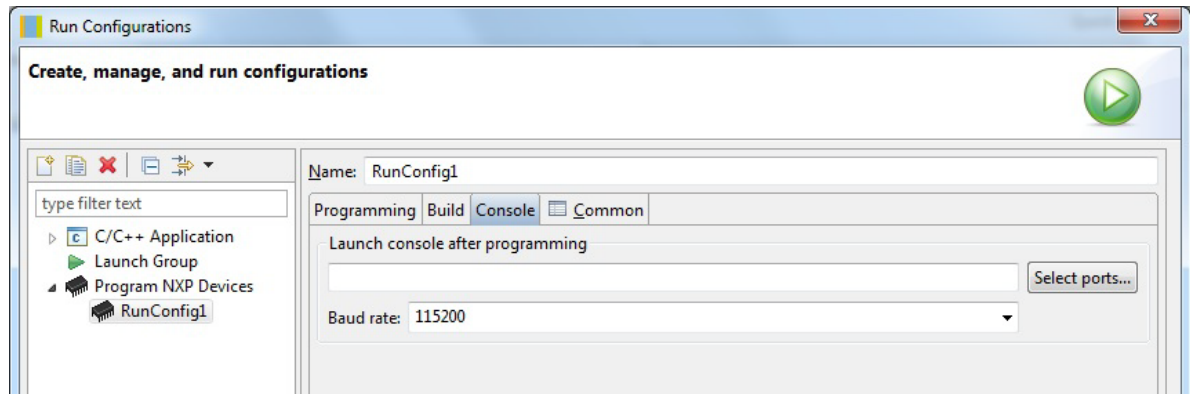
**Step 5** If required, you can add another Terminal tab by clicking the **Add Terminal** button (+ icon) on the toolbar in the top-right of the pane and then configuring this serial terminal as described from Step 2.



## D.2 Including a Terminal Tab in a Run Configuration

A Terminal tab can be assigned to a device in a Run Configuration (see [Section 3.4](#)) as follows (this procedure allows Terminal tabs to be assigned to multiple devices):

- Step 1** Within BeyondStudio for NXP, follow the menu path **Run>Run Configurations** to launch the **Run Configurations** dialogue box.
- Step 2** Under **Program NXP Devices** in the left panel, click on the Run Configuration to be modified.



- Step 3** On the **Console** tab on the right (see above), provide the following information:
- a)** From the first drop-down list, select the serial communications port(s) used for the connection(s) to the target device(s) to be associated with a Terminal tab.
  - b)** From the second drop-down list, select the baud rate for the connection(s) with the Terminal tab(s).

The configured Terminal tab(s) will automatically connect to the specified serial port(s) (also see Note 2 on page [47](#)).

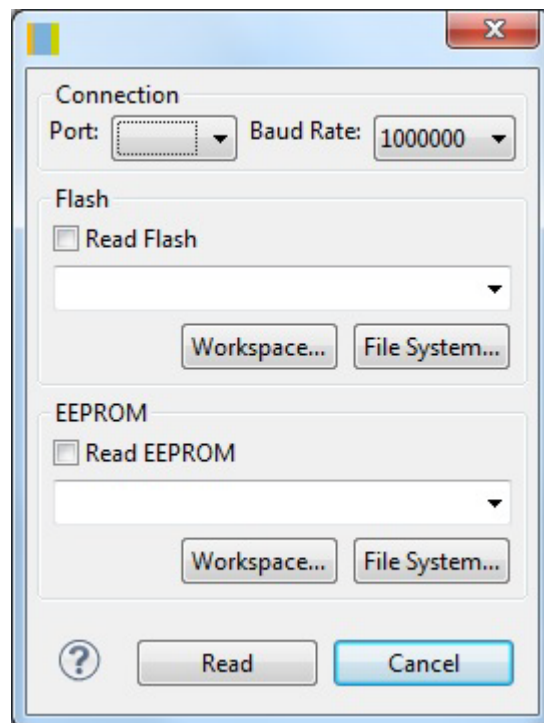
- Step 4** To save the configuration, click the **Apply** button.
- Step 5** To execute the Run Configuration, click the **Run** button, otherwise click the **Close** button.

## E. Reading the Contents of Flash Memory and EEPROM

BeyondStudio for NXP allows you to read the contents of the Flash memory and/or EEPROM of a JN516x device and dump the output to the file system on your PC.

To read the contents of the Flash memory and/or EEPROM of a connected JN516x device:

- Step 1** Ensure that the JN516x device to be accessed (which may be mounted on a board or module) is connected to a USB port of your PC.
- Step 2** Determine which serial communications port on your PC has been allocated to the USB connection - to identify the relevant port, refer to [Appendix A](#).
- Step 3** If not already running, launch BeyondStudio for NXP.
- Step 4** In BeyondStudio for NXP, follow the menu path **Devices>Read Device**. The following dialogue box will now appear.



- Step 5** In the dialogue box (shown above), configure the memory access as follows:

- **Connection** - Fill in the **Port** and **Baud Rate** fields as follows:
  - **Port** - Select the serial communications port used for the connection, as determined in Step 2
  - **Baud Rate** - Select the required data rate, in bps (normally, the default value of 1000000 should be used)

- **Flash** - To read from Flash memory, complete this section as follows:
  - Tick the **Read Flash** checkbox.
  - Navigate to the directory in which you wish the output file to be created (if this is in your workspace, use the **Workspace** button, otherwise use the **File System** button) and add the required filename to the end of the path
- **EEPROM** - To read from EEPROM, complete this section as follows:
  - Tick the **Read EEPROM** checkbox.
  - Navigate to the directory in which you wish the output file to be created (if this is in your workspace, use the **Workspace** button, otherwise use the **File System** button) and enter the required filename

Click the **Read** button. The dialogue box will now disappear and the contents of the specified memory will be read and output to the specified file(s).



## **Revision History**

<b>Version</b>	<b>Date</b>	<b>Comments</b>
1.0	13-June-2014	First release
1.1	17-Sept-2014	Updated branding
1.2	13-Mar-2015	Added installation of ZigBee plug-ins and other minor changes made

## **Important Notice**

**Limited warranty and liability** - Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

**Right to make changes** - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** - NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** - Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Export control** - This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

## **NXP Semiconductors**

For the contact details of your local NXP office or distributor, refer to:

**[www.nxp.com](http://www.nxp.com)**

For online support resources, visit the Wireless Connectivity TechZone:

**[www.nxp.com/techzones/wireless-connectivity](http://www.nxp.com/techzones/wireless-connectivity)**