HiWorkbench

User Guide

Issue **00B05**

Date **2015-04-30**

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Purpose

This document describes the functions and usage of the integrated development tool HiWorkbench. For details about the functions and usage of the C/C++ Development Toolkit (CDT) mentioned in this document, see the CDT help information in the HiWorkbench help system.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3110E	V3XX
Hi3712	V1XX
Hi3716C	V1XX
Hi3716C	V2XX
Hi3716M	V2XX
Hi3716M	V3XX
Hi3716M	V41X
Hi3716M	V42X
Hi3518	V1XX
Hi3521	V1XX
Hi3531	V1XX
Hi3500	V1XX
Hi3110E	V5XX
Hi3798C	V2XX

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
 WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
 CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 TIP	Provides a tip that may help you solve a problem or save time.
 NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 00B05 (2015-04-30)

This issue is the fifth draft release, which incorporates the following changes:

Hi3716M V420, Hi3716M V410, and Hi3798C V200 are supported.

Issue 00B04 (2015-03-10)

This issue is the fourth draft release, which incorporates the following changes:

The Hi3110E V500 is supported.

Issue 00B03 (2014-11-25)

This issue is the third draft release, which incorporates the following changes:

Chapter 2 Environment Preparations

Section 2.5 is added.

Issue 00B02 (2014-09-05)

This issue is the second draft release, which incorporates the following changes:

Chapter 6 FAQs

Section 6.7 is added.

Issue 00B01 (2014-05-15)

This issue is the first draft release.

Contents

Figures

1 Overview

The HiWorkbench is a professional software development solution designed based on the Linux system and bare board embedded system. Its functions cover various development phases from code porting development to application and bare board program debugging, including the following:

- C/C++ code editing
- Remote or local compilation of C/C++ code
- Debugging of Linux applications, bare board programs, and local core files

Before using the HiWorkbench, ensure that the Java runtime environment (JRE) or Java development kit (JDK) V1.6 (or a later) has been installed. You can obtain the JRE or JDK from the following website:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

If you need to use the local compilation function of the HiWorkbench, ensure that the CodeSourcery tool chain has been installed. You can obtain the tool chain from the following website:

https://sourcery.mentor.com/GNUToolchain/release2450

Figure 1-1 shows the startup GUI of the HiWorkbench.

Figure 1-1 Startup GUI

The main GUI is displayed after the startup GUI. It consists of three parts: menu bar, toolbar, and perspective view (the red rectangle region in Figure 1-2). Figure 1-2 shows the main GUI for the C/C++ code editing function, and Figure 1-3 shows the main GUI for the debugging function.

Figure 1-2 C/C++ perspective view

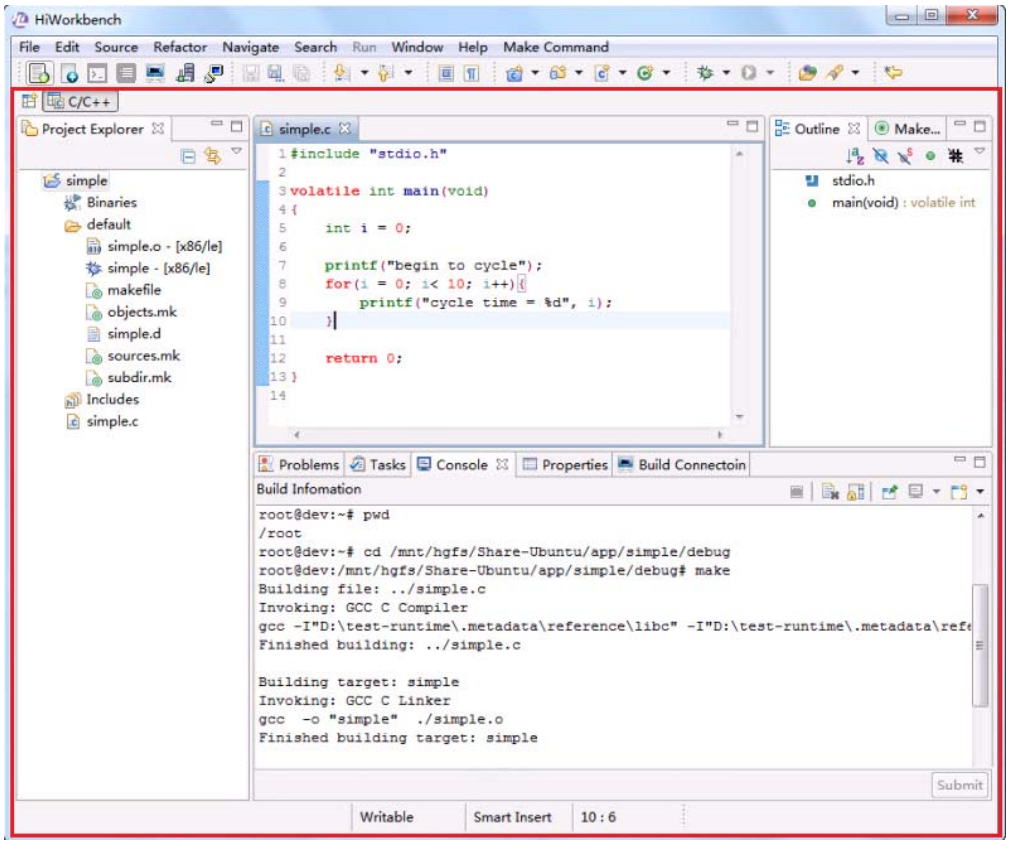
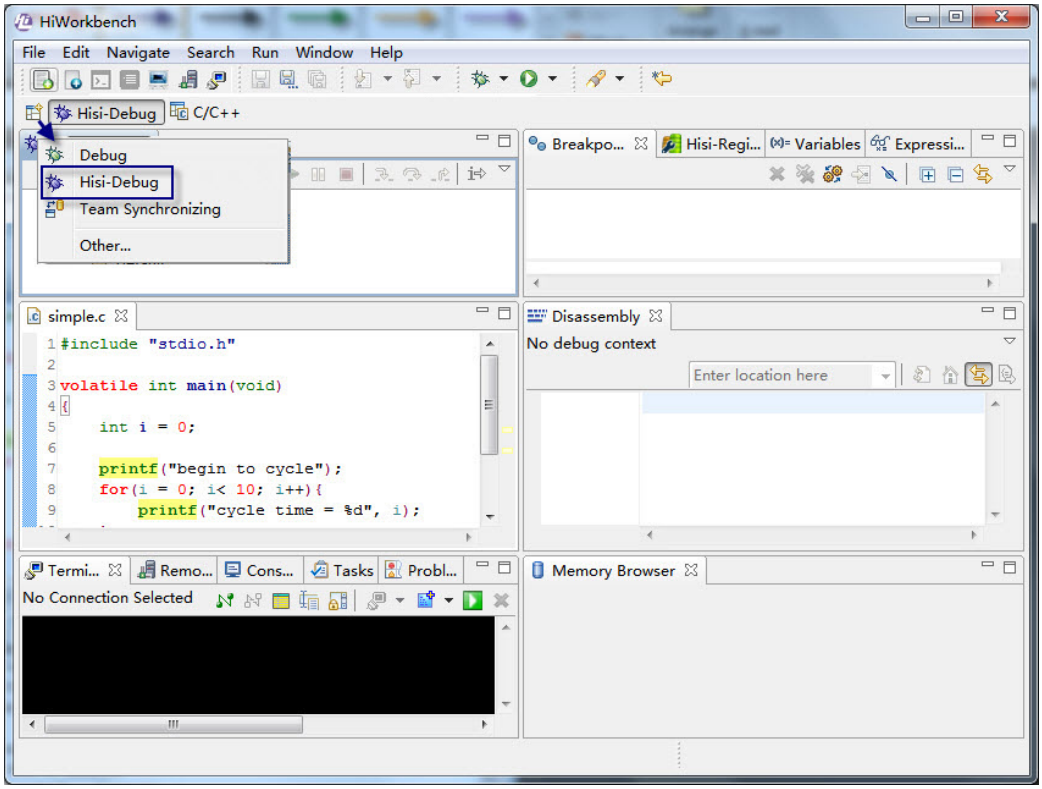


Figure 1-3 Hisi-Debug perspective view



2 Environment Preparations

2.1 Mapping the Network Drive

The HiWorkbench allows you to remotely develop or debug the source code on a Linux server. If you need to use this function, map the directory for storing the source code to the local as a network drive.

NOTE

For details about how to map a network drive, see the OS help information.

2.2 Preparing the Environment for JTAG Debugging

Make the following preparations before debugging bare board programs:

- Hardware: target board, Hi-ICE emulator (with JTAG debugging cables and USB cables)
- Software: Hi-ICE driver, source code and binary files to be debugged

2.3 Preparing the Environment for Debugging Linux Applications

Make the following preparations before debugging Linux applications:

- Hardware: target board, Ethernet cables
- Software: source code and binary files to be debugged

2.4 Installing the Local Tool Chain

To install the local tool chain, perform the following steps:

Step 1 Download the CodeSourcery tool chain from https://sourcery.mentor.com/GNUToolchain/release2450 and install it by using the default configurations.

Step 2 Right-click **Computer**, and choose **Properties** > **Advanced system settings**. Click **Environment Variables**, select **PATH**, click **Edit**, and enter the installation path of the tool chain (**xxxxxx\CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_GNU_Linux\bin**).

Step 3 Change **cs-make** and **cs-rm** to **make** and **rm** in the installation path of the tool chain (**xxxxxx\CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_GNU_Linux\bin**).

Step 4 Install the Linux shell tool package required for Windows compilation. Download **coreutils-5.3.0.exe** and **sed-4.2.1-setup.exe** from http://ftp.gnu.org/gnu, and install them by using default configurations. The two programs are installed in the same path by default.

Step 5 Write **XXX\GnuWin32\bin** to the **PATH** environment variable. For details, see step 2.

Step 6 Restart the system to complete the installation.

----End

2.5 Installing the Hi-ICE Driver

To install the Hi-ICE driver, perform the following steps:

Step 1 Decompress the Hi-ICE driver package **CDM v2.10.00 WHQL Certified.zip** in the SDK.

Step 2 Connect the Hi-ICE to the USB port of the PC.

Ensure that the USB read and write functions of the PC are normal.

Step 3 Install the Hi-ICE driver according to the instructions.

Figure 2-1 shows the device status in the **Device Manager** when the driver is not installed properly, and Figure 2-2 shows the device status when the driver is installed properly.

Figure 2-1 Device Manager when the Hi-ICE driver is not installed properly

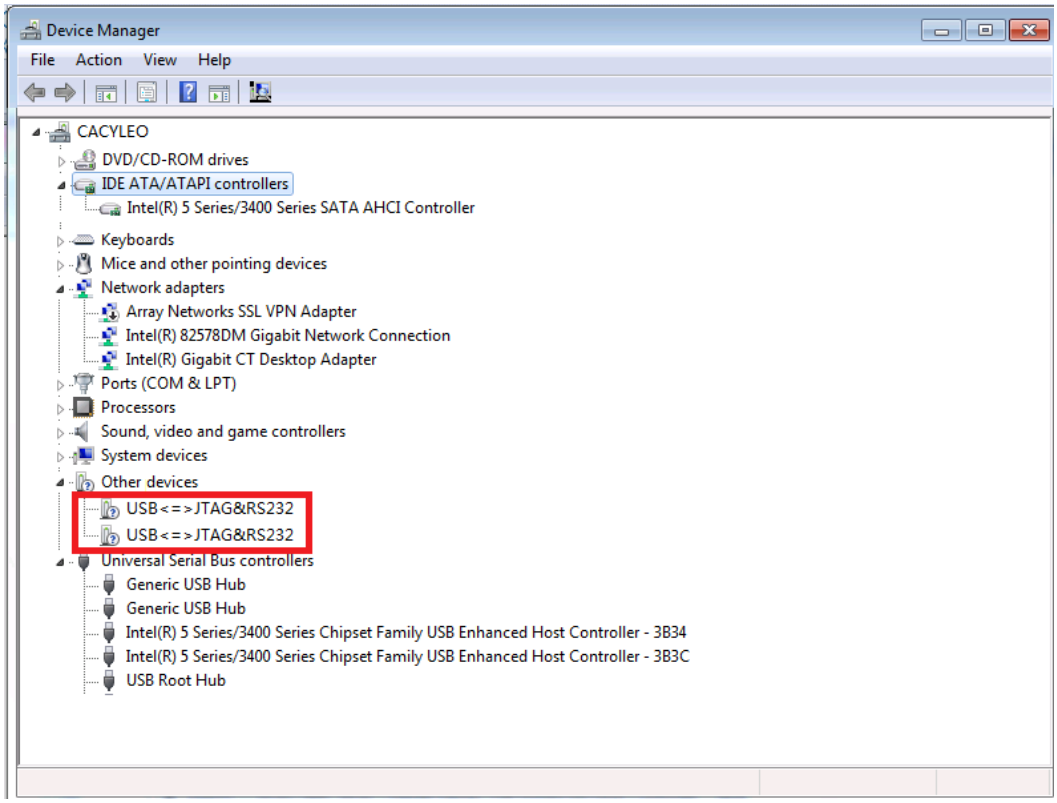
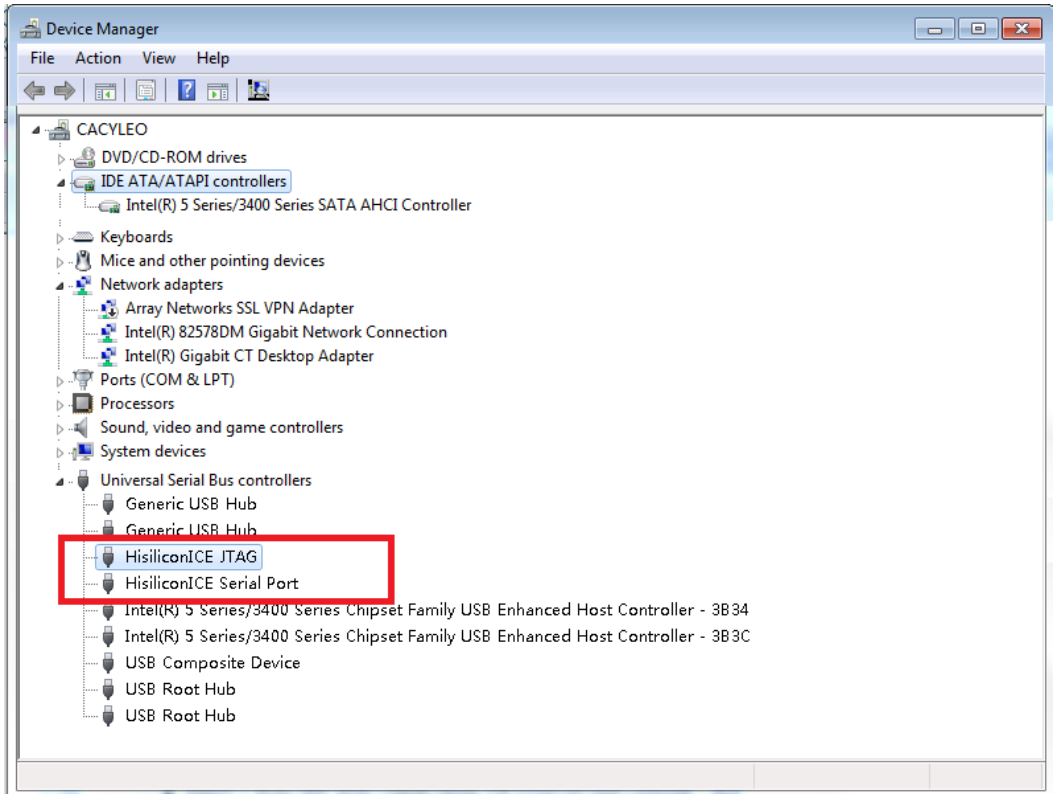


Figure 2-2 Device Manager when the Hi-ICE driver is installed properly



----End

3 Getting Started

This chapter describes how to develop and debug applications by using the HiWorkbench.

3.1 Creating a Project

3.1.1 Creating a Remote Project

To create a remote project, perform the following steps:

Step 1 Choose **File** > **New** on the menu bar, and choose **Hisilicon** > **Project With Existing Code** in the displayed **New** dialog box, as shown in Figure 3-1; or right-click **Project Explore**, and choose **New** > **Project With Existing Code**, as shown in Figure 3-2.

Figure 3-1 Creating a remote project by using the File menu

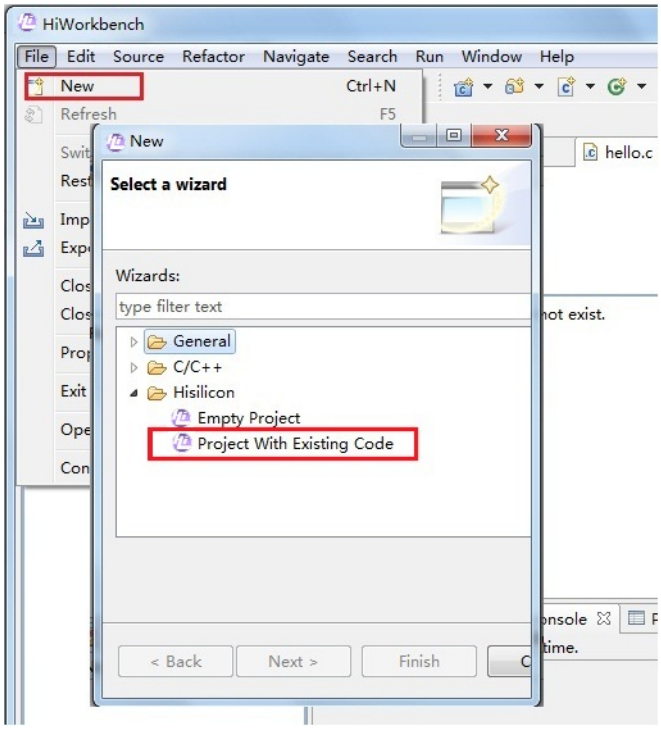
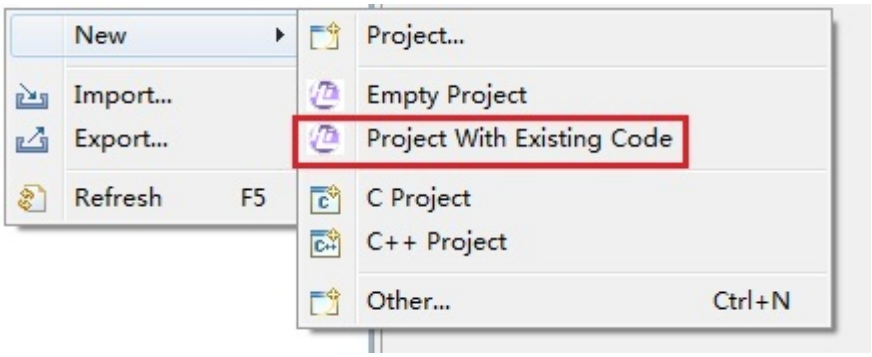
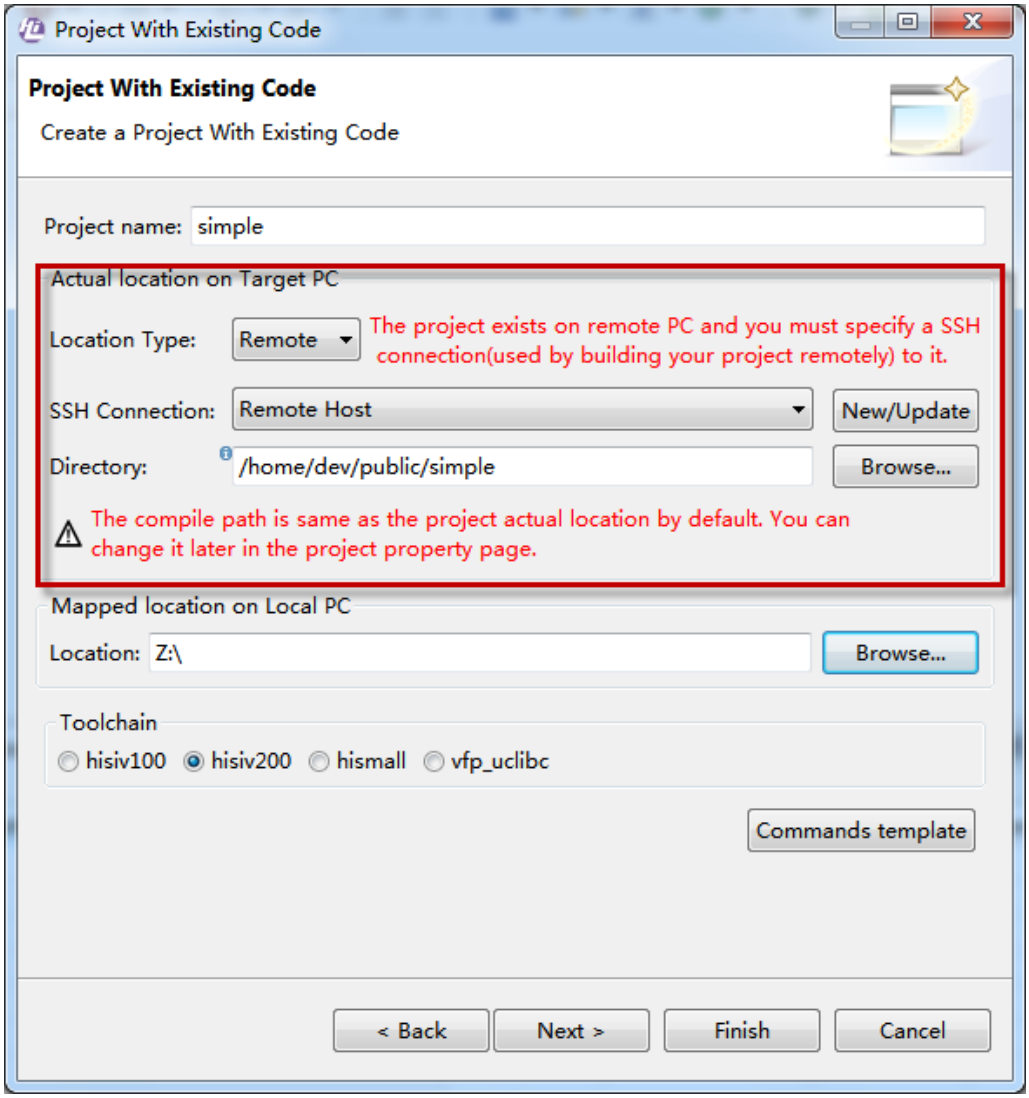


Figure 3-2 Creating a remote project by using the shortcut menu



Step 2 Enter a project name, set **Location Type** to **Remote**, select a secure shell (SSH) connection to a remote server, and specify the compilation command execution path of the project to be created on the remote server in the displayed dialog box, as shown in Figure 3-3. If the required remote server SSH connection is not in the **SSH Connection** drop-down list, create an SSH connection. See step 3.

Figure 3-3 Creating a remote project



Step 3 (Optional) Click **New/Update** to create an SSH connection to a remote server. Enter the following information in the displayed dialog box shown in Figure 3-4, and click **Finish**. An SSH connection is established.

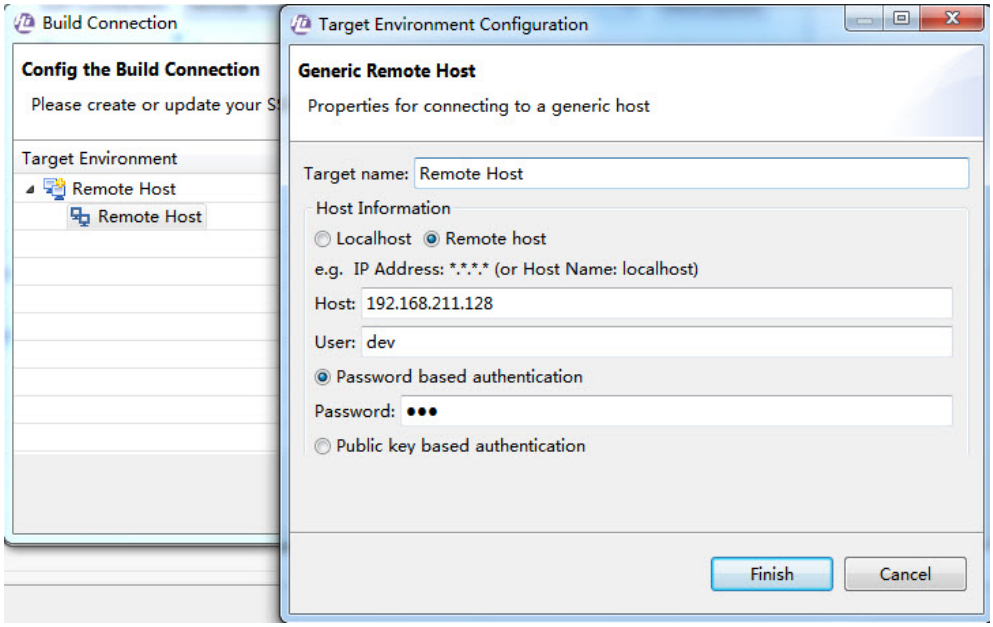
Target Name: name of the SSH connection that is displayed in the **SSH Connection** drop-down list in Figure 3-3

Host: IP address for the server for storing the source code

User: account for logging in to the server

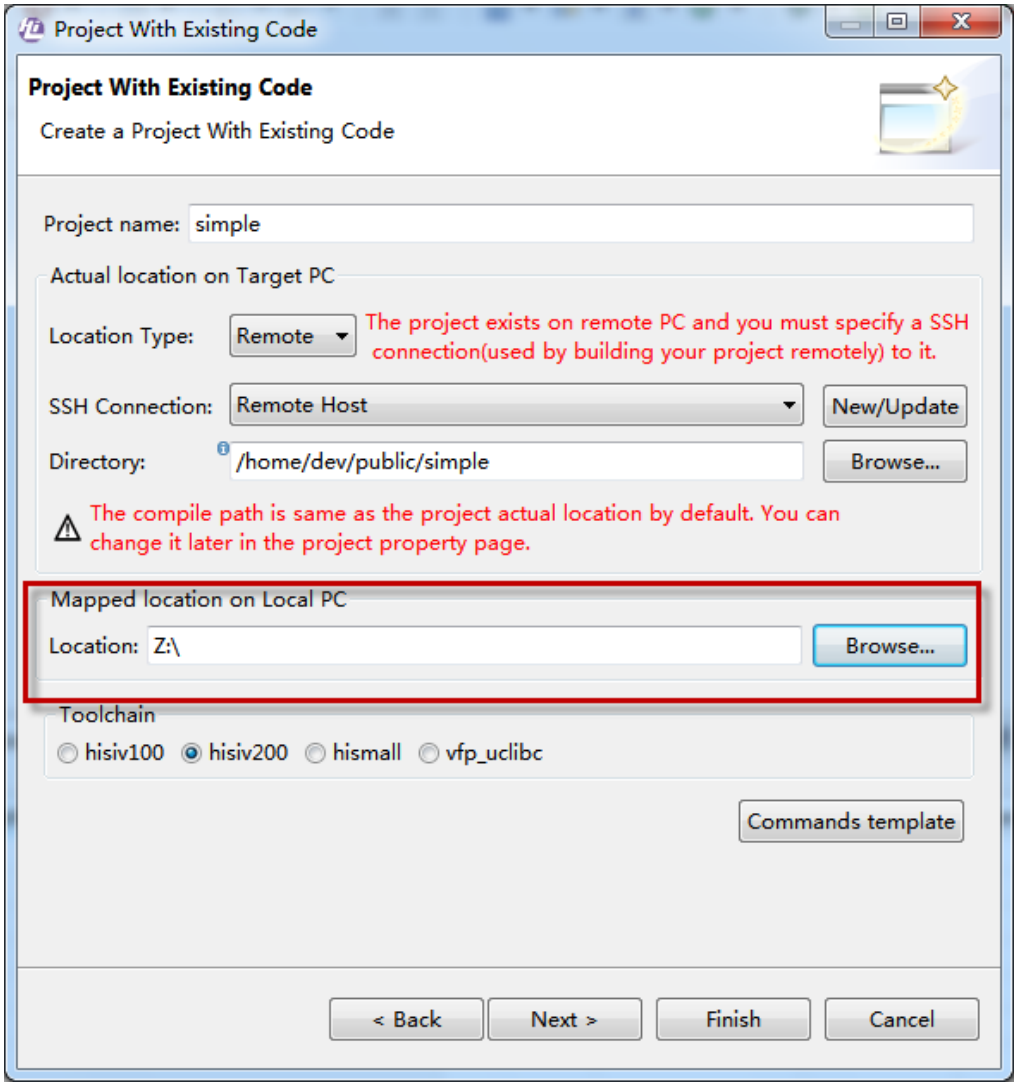
Password: password of the account. **Password based authentication** needs to be selected.

Figure 3-4 Creating/updating an SSH connection



Step 4 Specify the position for mapping the remote project to the local. Map the directory for storing the source code to the local as a network drive first. See Figure 3-5.

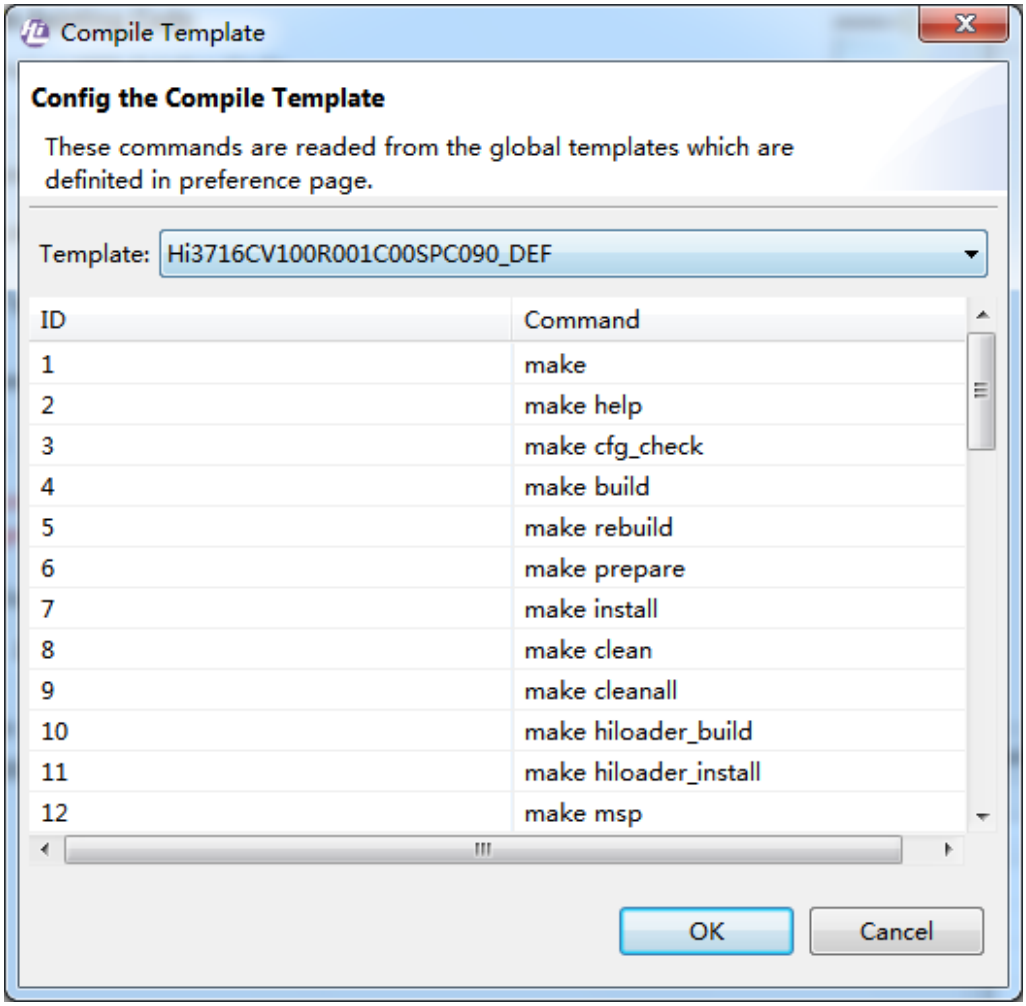
Figure 3-5 Specifying the mapping position



Step 5 Select the tool chain for compiling the project.

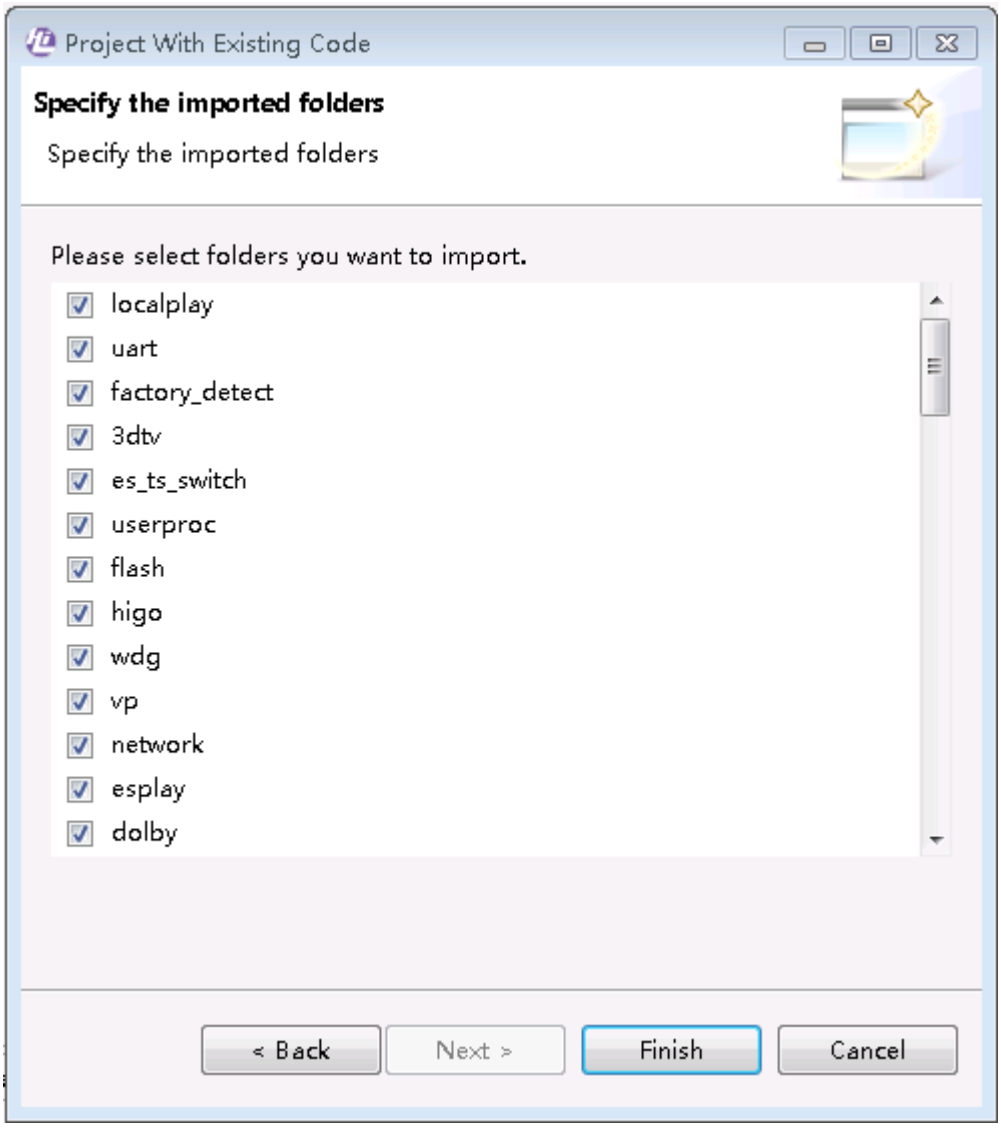
Step 6 (Optional) Click **Commands template**, and select a compilation command template. See Figure 3-6.

Figure 3-6 Selecting a template



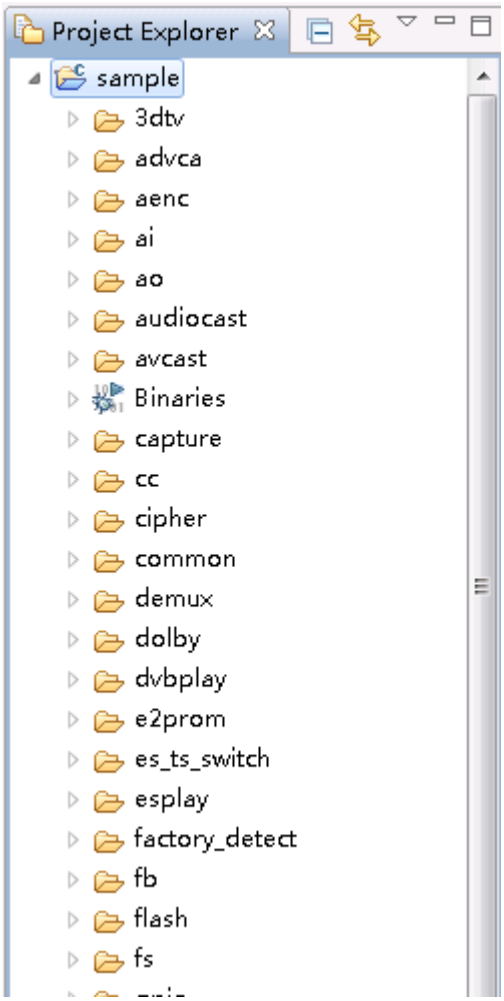
Step 7 Click **Next**. The dialog box shown in Figure 3-7 is displayed. Select the top directories of the project to be imported (all top directories are selected by default). This step prevents unnecessary directories from being imported, which may result in tool performance deterioration.

Figure 3-7 Selecting directories to be imported



Step 8 Click **Finish**. A remote project is created and displayed in the **Project Explorer** view, as shown in Figure 3-8.

Figure 3-8 Created project in Project Explore



For details about how to create a project, see section 5.1 "Creating a Project."

----End

3.1.2 Creating a Remote Empty Project

To create a remote empty project, perform the following steps:

Step 1 Choose **File** > **New** on the menu bar, and choose **Hisilicon** > **Empty Project** in the displayed **New** dialog box, as shown in Figure 3-9; or right-click **Project Explore**, and choose **New** > **Empty Project**, as shown in Figure 3-10.

Figure 3-9 Creating a remote empty project by using the File menu

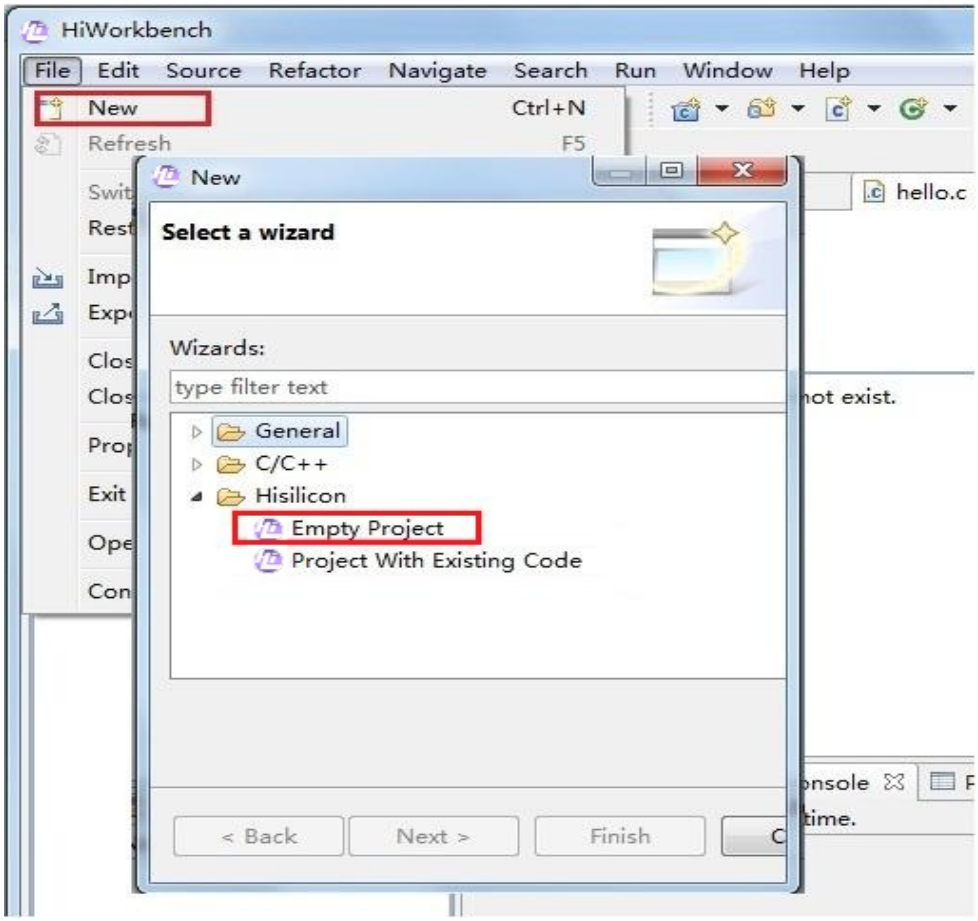
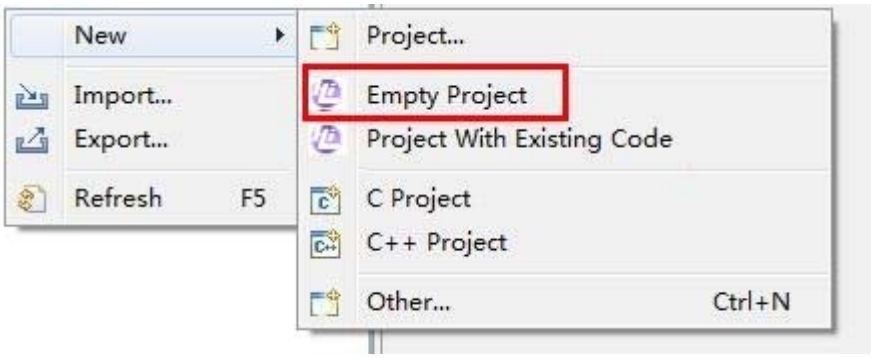
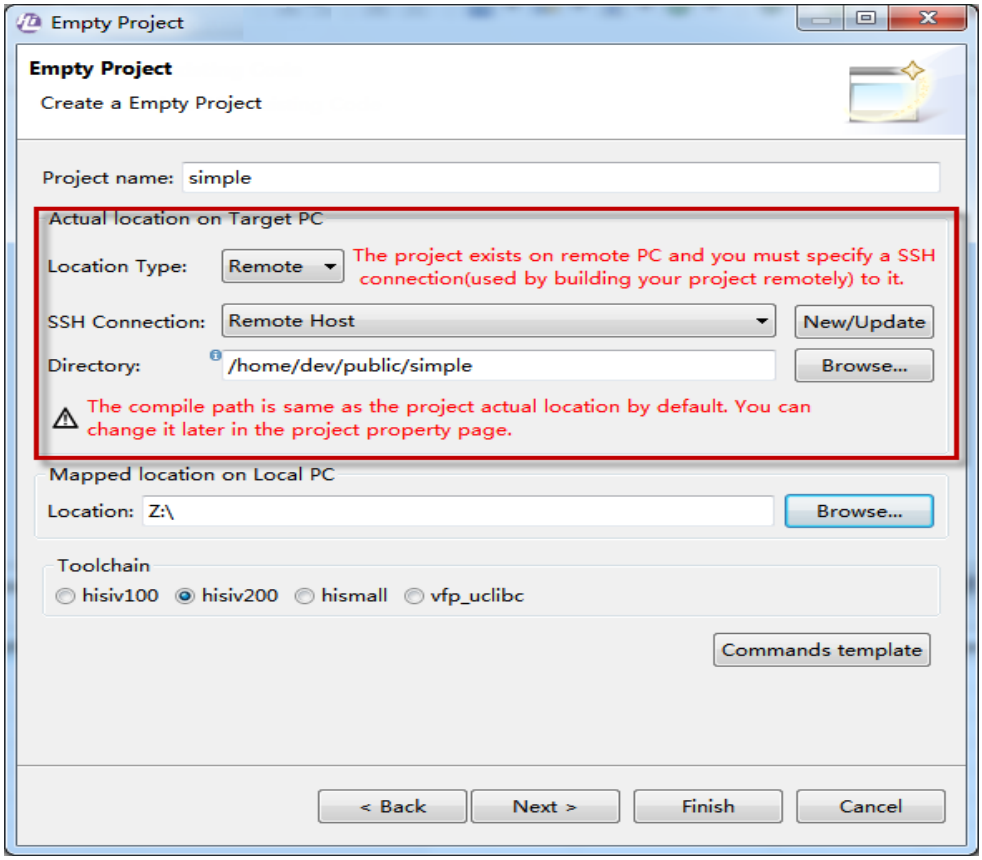


Figure 3-10 Creating a remote empty project by using the shortcut menu



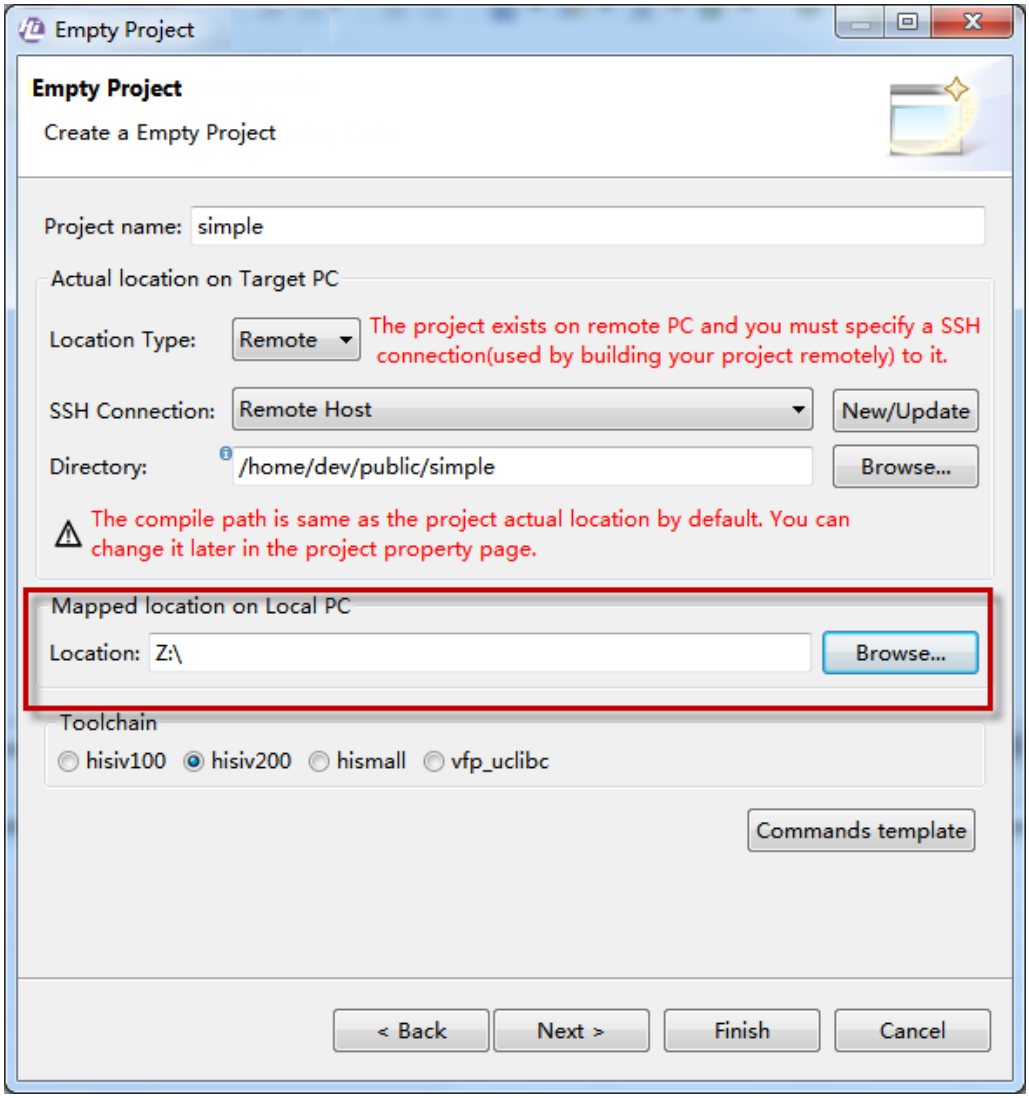
Step 2 Enter a project name, set **Location Type** to **Remote**, select an SSH connection of a remote server, and specify the compilation command execution path of the project to be created on the remote server in the displayed dialog box, as shown in Figure 3-11. If the required remote server SSH connection is not in the **SSH Connection** drop-down list, create an SSH connection. For details, see step 3 in section 3.1.1 "Creating a Remote Project."

Figure 3-11 Specifying the actual location on the remote host



Step 3 Specify the position for mapping the remote project to the local. Map the directory for storing the source code to the local as a network drive first. See Figure 3-12.

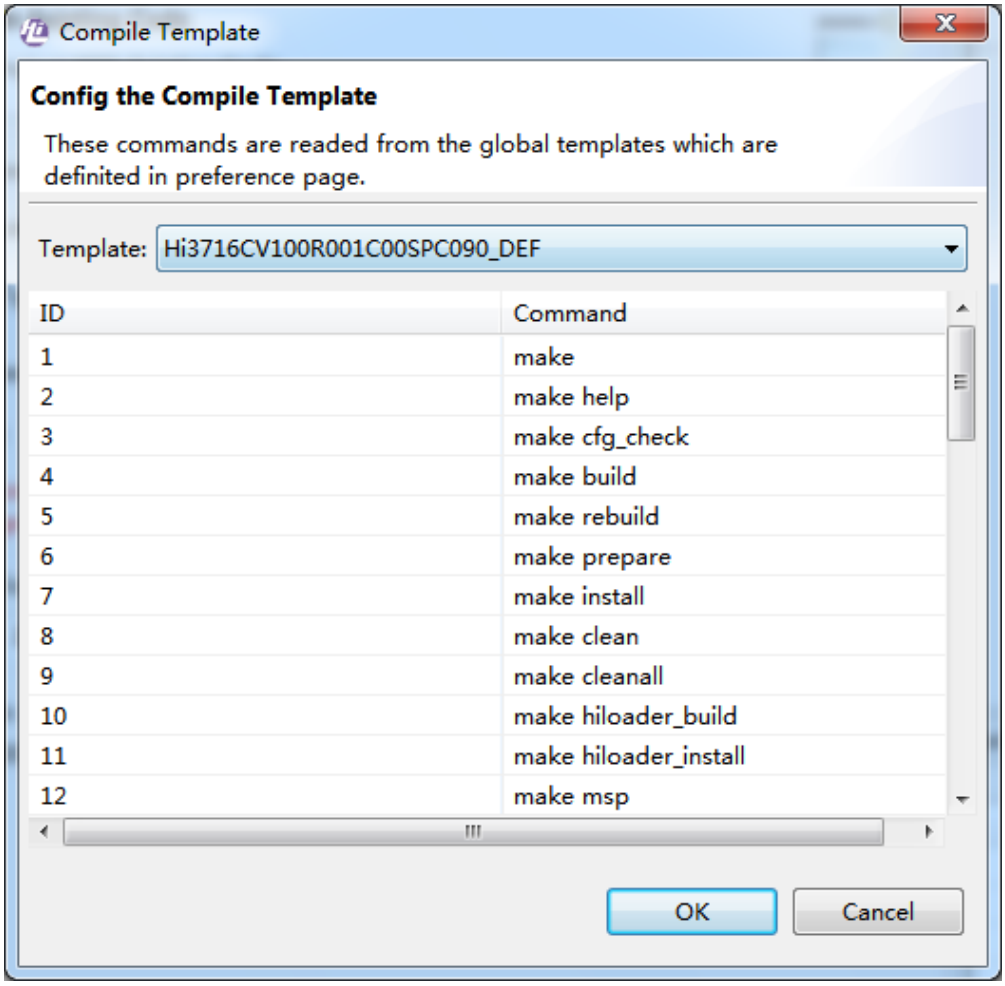
Figure 3-12 Specifying the mapping position



Step 4 Select the tool chain for compiling the project.

Step 5 (Optional) Click **Commands template**, and select a compilation command template. See Figure 3-13.

Figure 3-13 Selecting a template



Step 6 Click **Finish**. A remote empty project is created and displayed in the **Project Explorer** view.

For details about how to create a project, see section 5.1 "Creating a Project."

----End

3.1.3 Creating a Local Project

To create a local project, perform the following steps:

Step 1 Choose **File** > **New** on the menu bar, and choose **Hisilicon** > **Project With Existing Code** in the displayed **New** dialog box, as shown in Figure 3-14; or right-click **Project Explore**, and choose **New** > **Project With Existing Code**, as shown in Figure 3-15.

Figure 3-14 Creating a local project by using the File menu

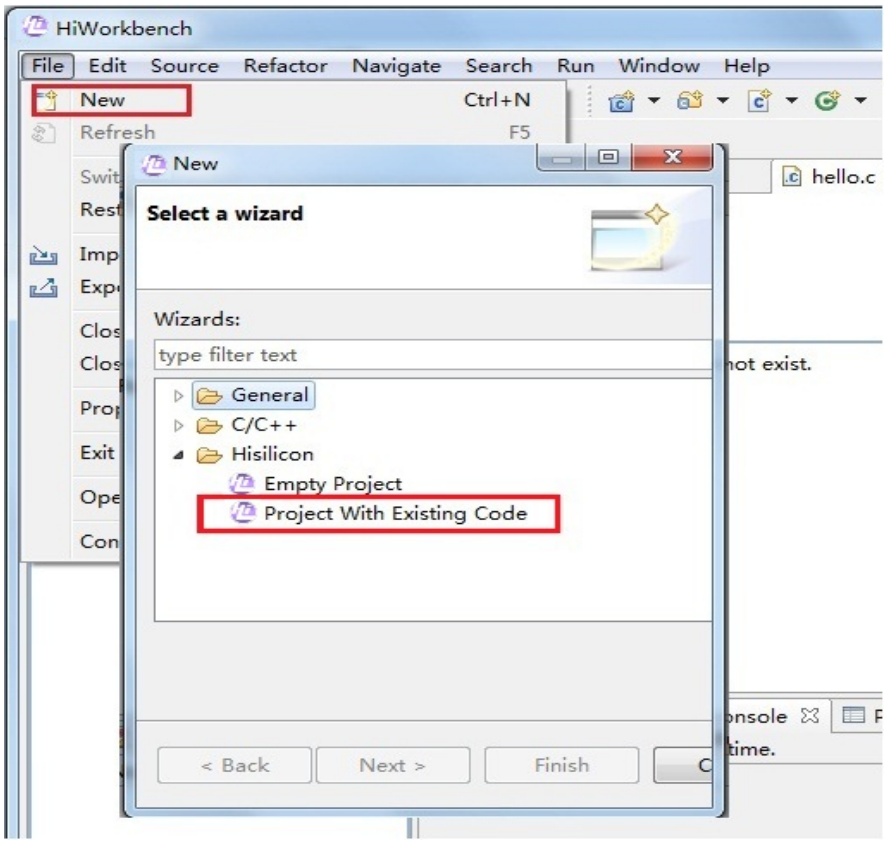
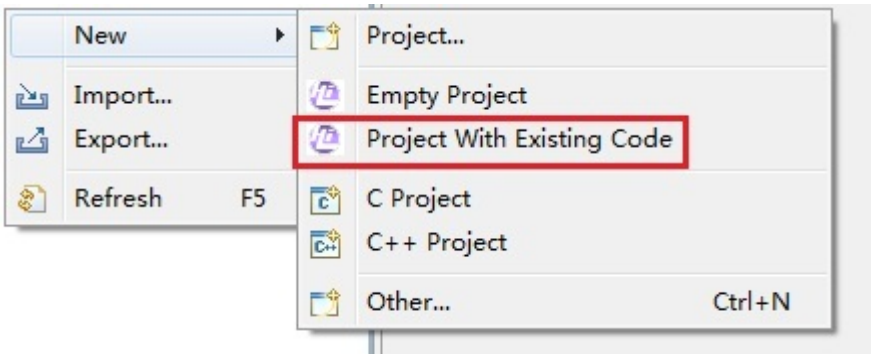
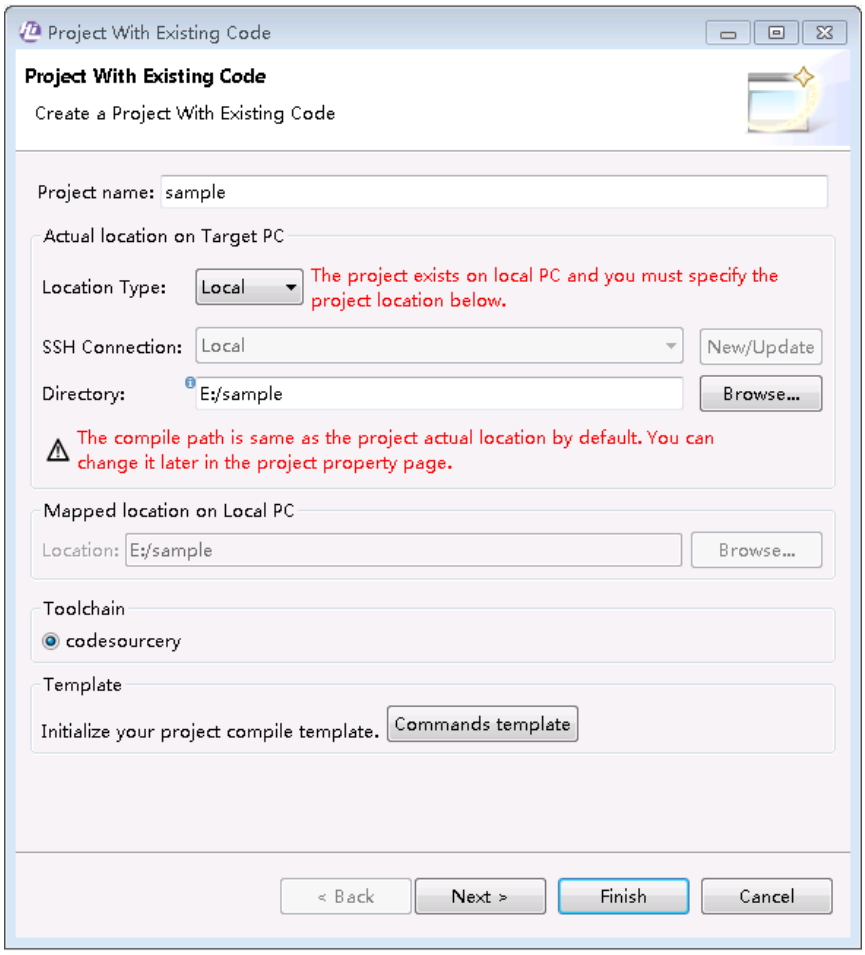


Figure 3-15 Creating a local project by using the shortcut menu



Step 2 Enter a project name, set **Location Type** to **Local**, and specify the absolute path for the project on the local PC in the displayed dialog box. See Figure 3-16.

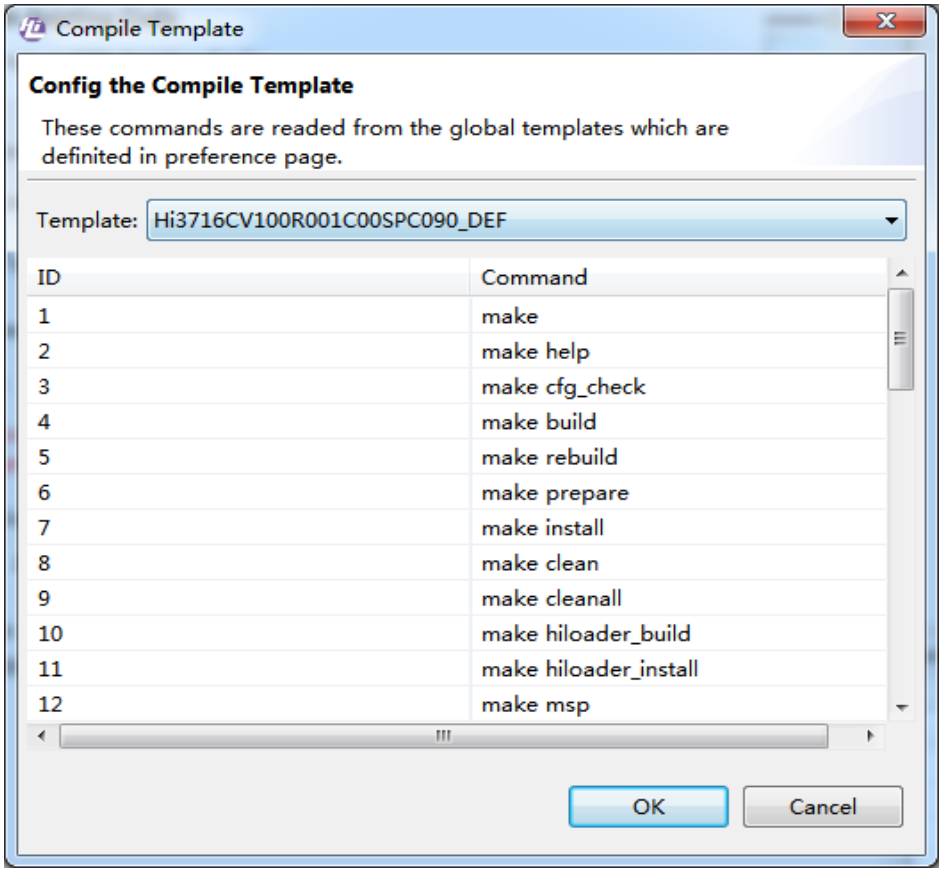
Figure 3-16 Specifying the actual position on the local PC



Step 3 Select the tool chain for compiling the project (only the CodeSourcery can be used to compile local projects).

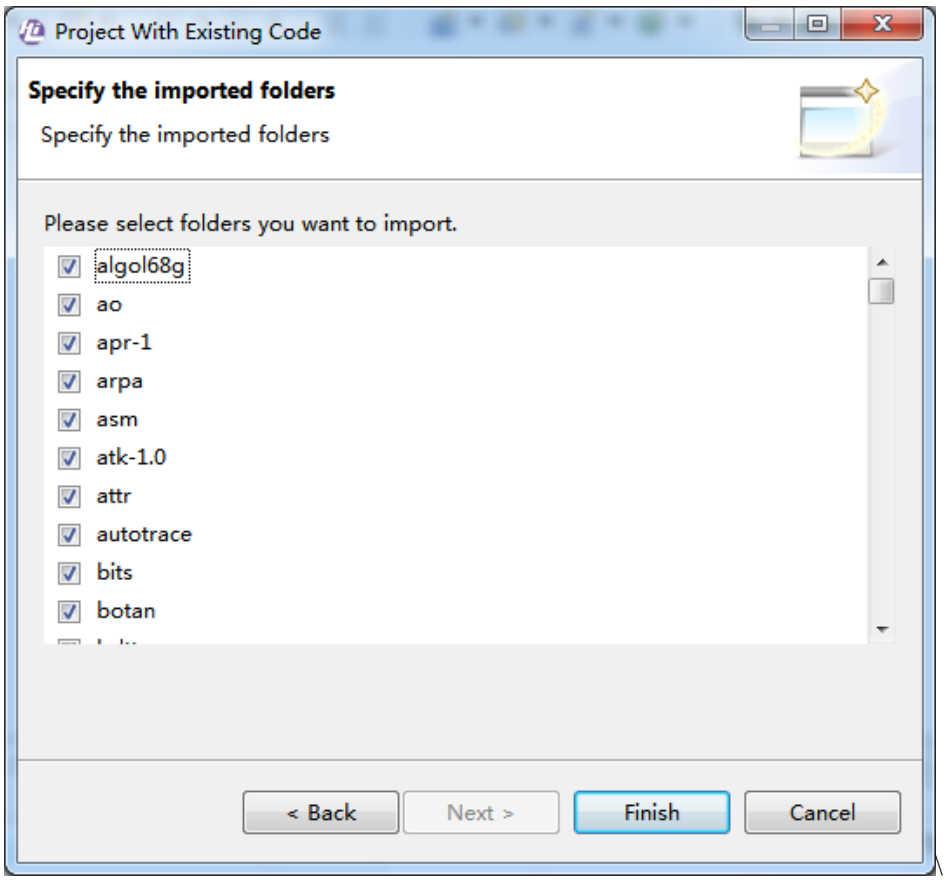
Step 4 (Optional) Click **Commands template**, and select a compilation command template. See Figure 3-17.

Figure 3-17 Selecting a template



Step 5 (Optional) Click **Next**, and select the top directories to be imported. Only top directories in the project are visible, and all of them are selected by default. See Figure 3-18.

Figure 3-18 Selecting directories to be imported



Step 6 Click **Finish**. A local project is created and displayed in the **Project Explorer** view.

For details about how to create a project, see section 5.1 "Creating a Project."

----End

3.1.4 Importing a Project

To import a project, perform the following steps:

Step 1 Choose **File** > **Import** on the menu bar, as shown in Figure 3-19; or right-click **Project Explore**, and choose **Import**, as shown in Figure 3-20.

Figure 3-19 Importing a project by using the File menu

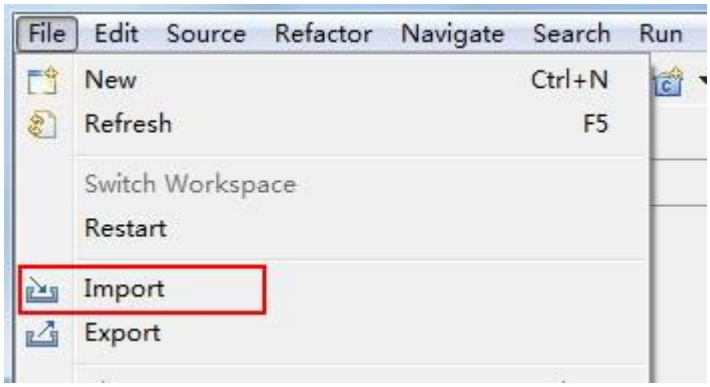
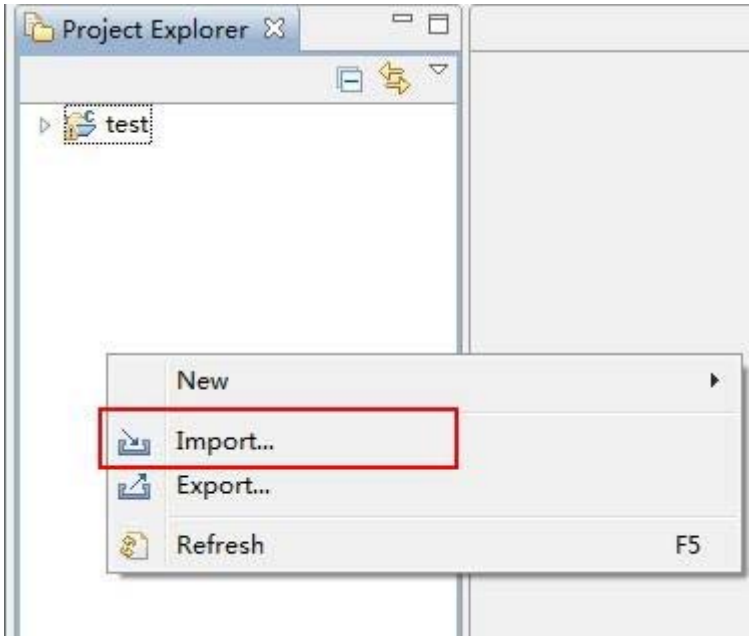
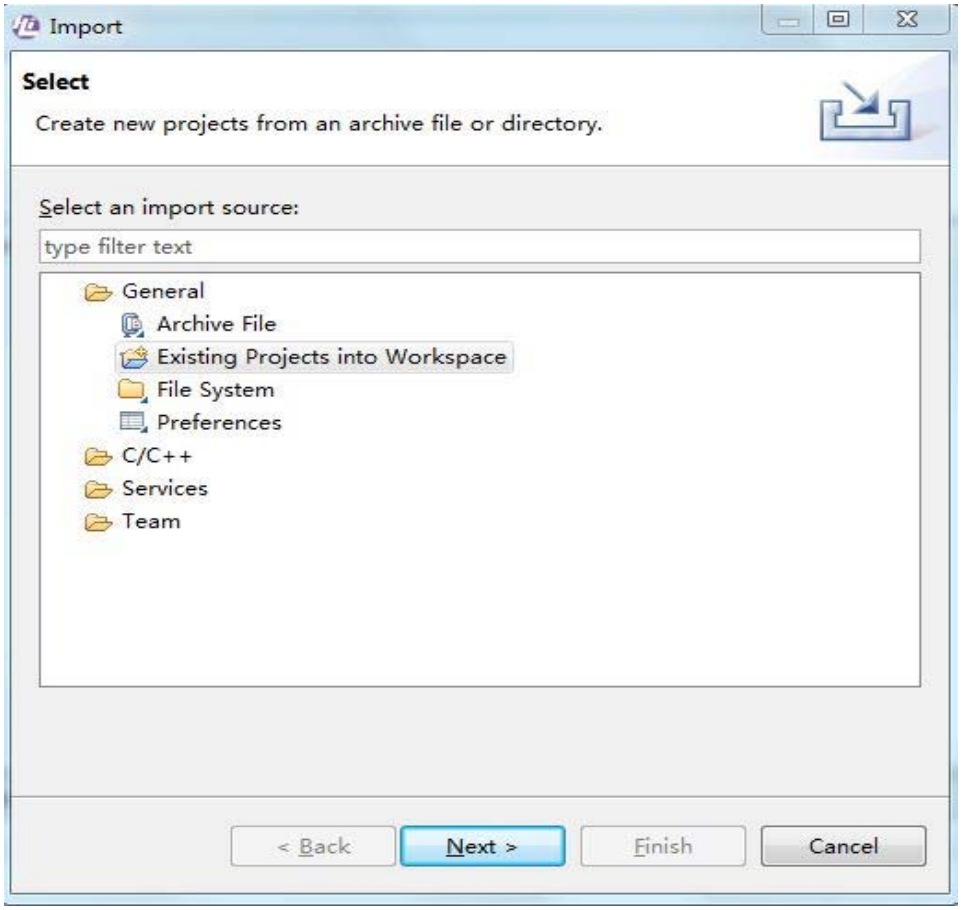


Figure 3-20 Importing a project by using the shortcut menu



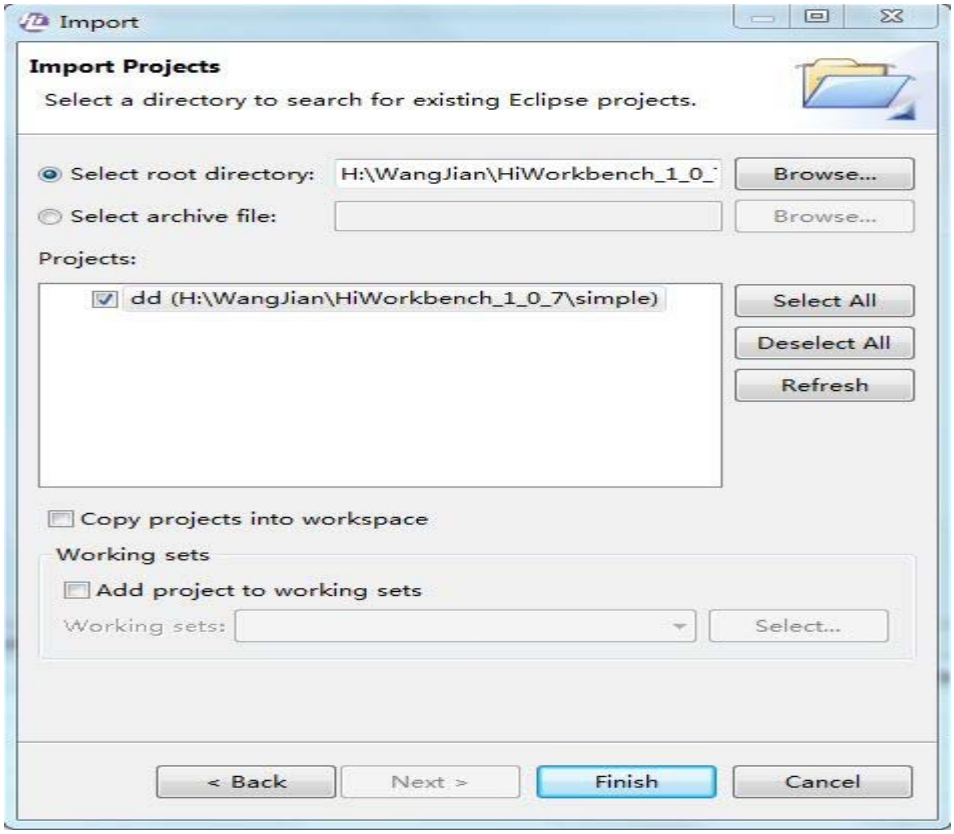
Step 2 Choose **General** > **Existing Projects into Workspace**, and click **Next**, as shown in Figure 3-21.

Figure 3-21 Import dialog box



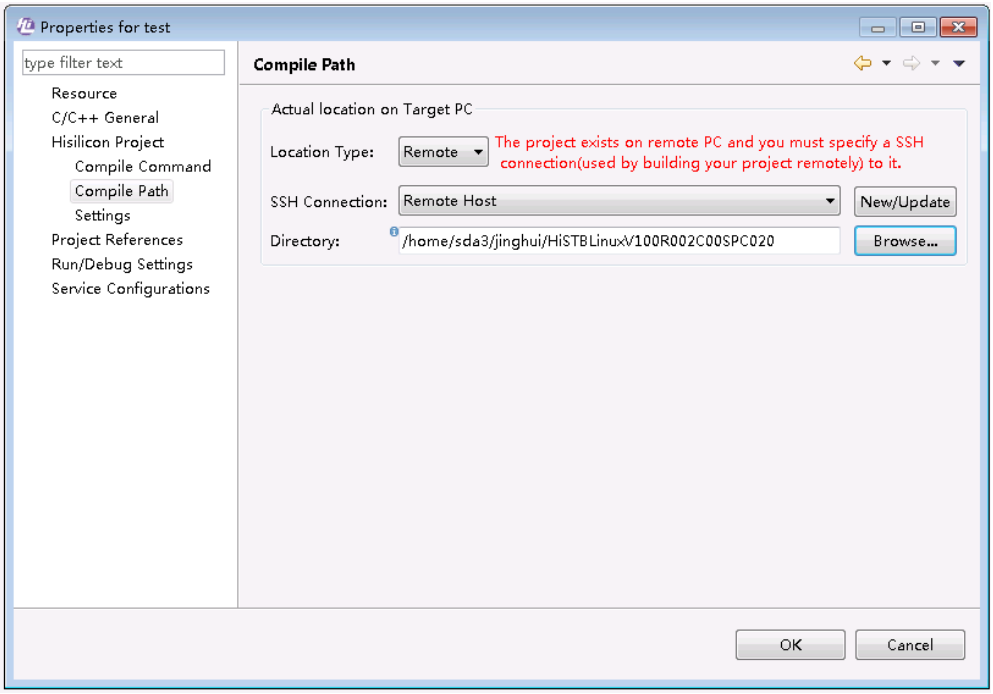
Step 3 Specify the path for storing the project, select the project to be imported, and click **Finish**. The project is displayed in the **Project Explorer** view.

Figure 3-22 Selecting the project to be imported



Step 4 Right-click the project to be imported in the **Project Explorer**, and choose **Properties** from the shortcut menu. Then choose **Hisilicon Project** > **Compile Path**, specify **SSH Connection**, and set the compilation path, as shown in Figure 3-23. If the required SSH connection is not in the **SSH Connection** drop-down list, create an SSH connection. For details, see step 3 in section 3.1.1 "Creating a Remote Project." If the project to be imported is a remote project, reconfigure the compilation path; if it is a local project, you do not need to set the tool chain because the HiWorkbench automatically uses the CodeSourcery for compilation.

Figure 3-23 Setting the compilation path



----End

3.2 Compiling a Project

The HiWorkbench supports both remote compilation and local compilation. The compilation mode depends on whether the project is a remote one or a local one.

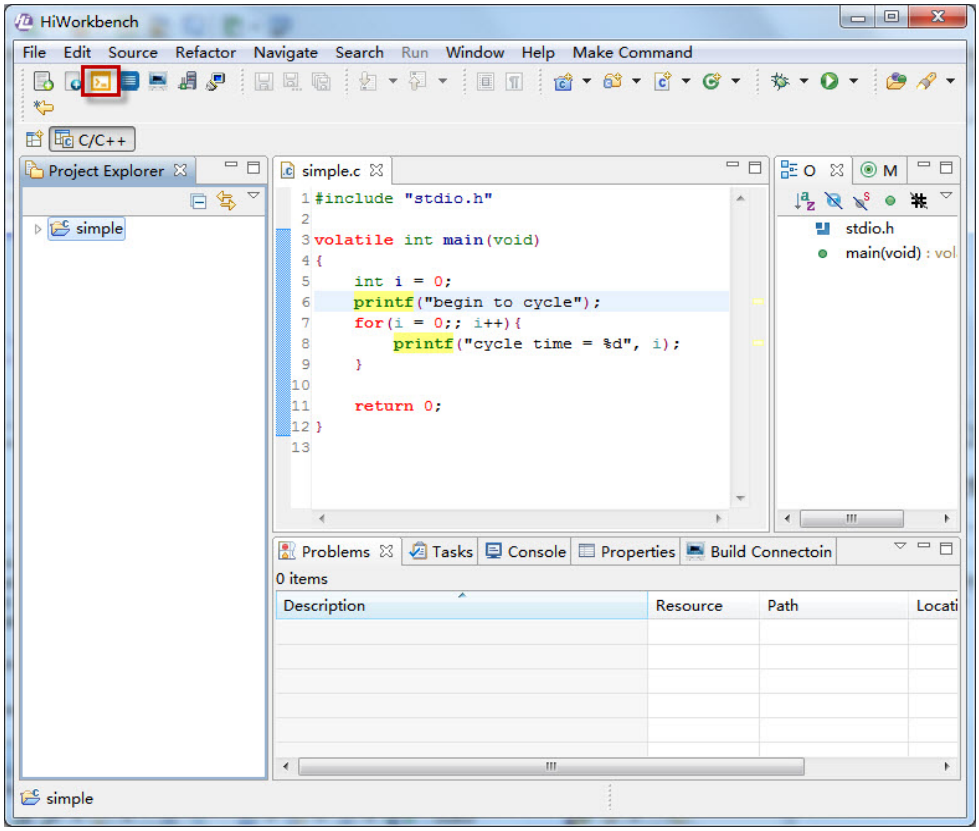
For a remote project, the HiWorkbench uses remote compilation by logging in to the remote compilation server over SSH connection and running compilation commands in the specified directory. Before compiling a project, ensure that the SSH connection and compilation path are correctly configured. You can set the SSH connection and compilation path when creating the project, or modify them on the project properties page. For details about how to modify the compilation path for a created project, see section 5.8 "Compilation Path Update." Each compilation command is displayed in the **Make Command** menu as a sub menu. When you select a sub menu, the corresponding command is executed on the remote host over the SSH connection.

For a local project, the HiWorkbench uses local compilation. The CodeSourcery tool chain (you need to download and install it) is used by default.

To compile a project, perform the following steps:

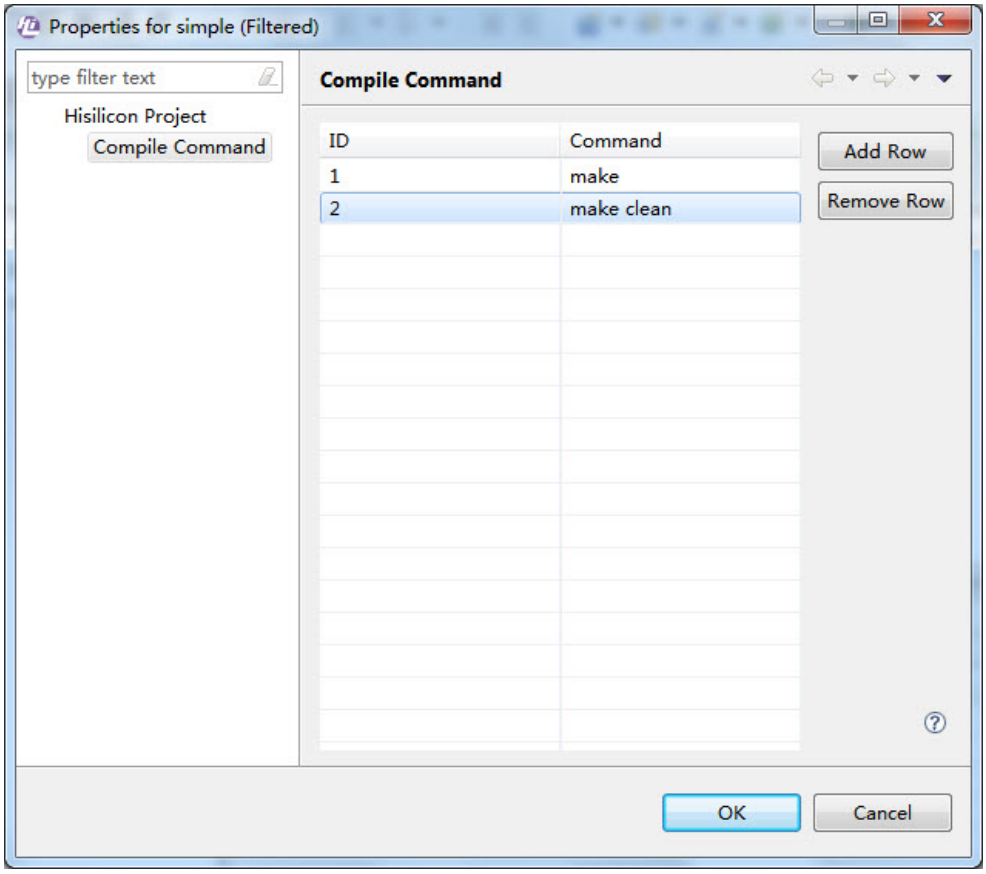
Step 1 Select a created project, and click the image icon in the red rectangle shown in Figure 3-24.

Figure 3-24 Compilation command button



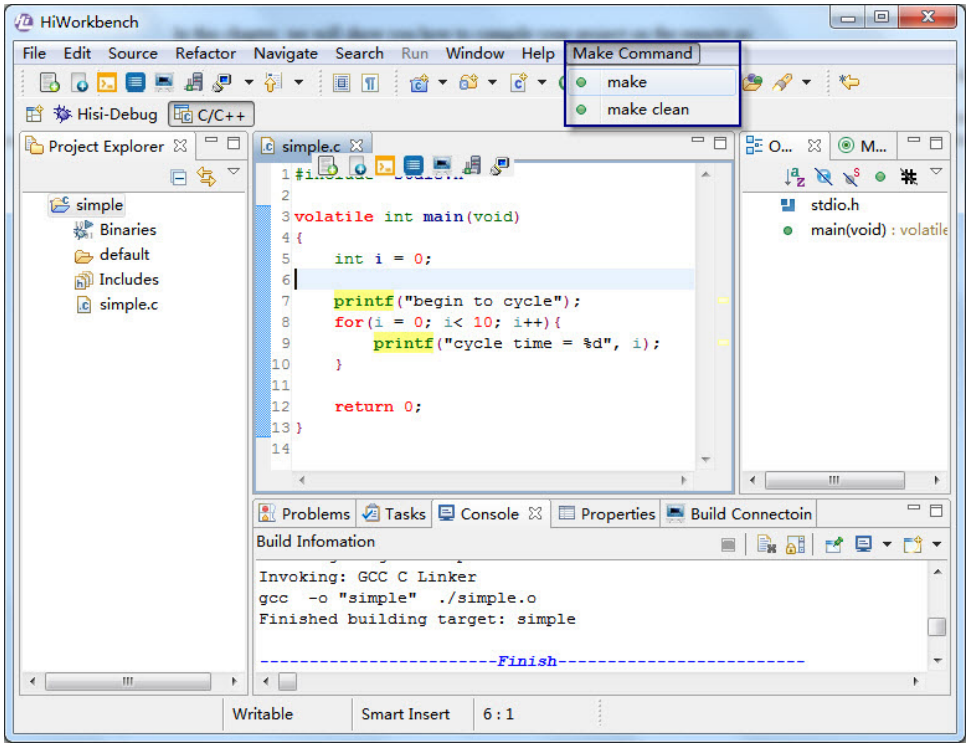
Step 2 Add compilation commands in the displayed dialog box, and click **OK** to save the commands. See Figure 3-25.

Figure 3-25 Adding compilation commands



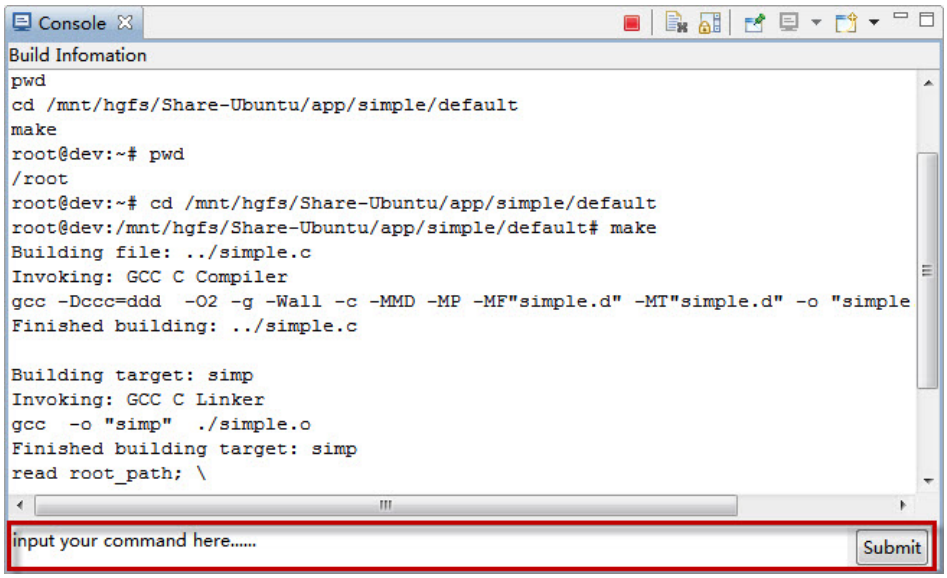
Step 3 Select a project, and choose the compilation command to be executed in the **Make Command** menu. The HiWorkbench then sends the compilation command to the remote compilation server. See Figure 3-26.

Figure 3-26 Choosing a compilation command



You can enter a command or other information during compilation in the red rectangle shown in Figure 3-27.

Figure 3-27 Entering a command



You can click  to stop the compilation.

----End

3.3 Debugging a Project

This section describes how to debug a project.

3.3.1 Debugging a Linux Application

After the Linux application is compiled and the binary file for debugging is generated, you can debug the source code on the board as follows:

Step 1 Right-click the project name, and choose **Debug As** > **Debug Configurations**, as shown in Figure 3-28; or choose **Debug Configurations** from the drop-down list of the debug button on the toolbar, as shown in Figure 3-29.

Figure 3-28 Opening the Debug Configurations dialog box by using the shortcut menu

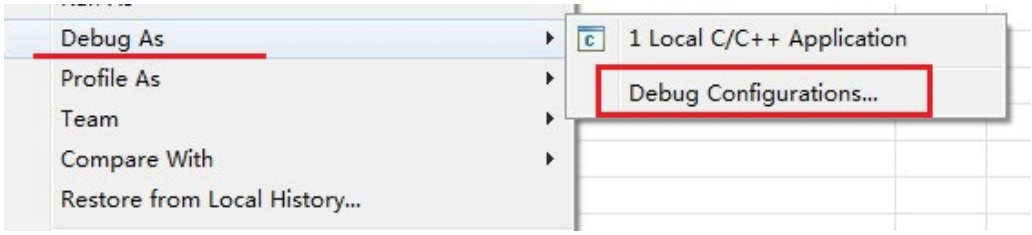
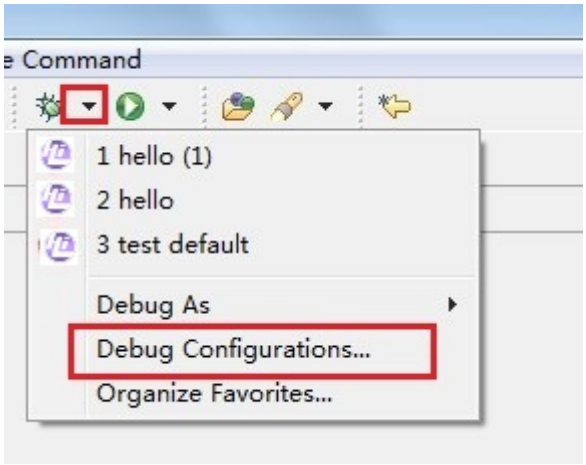
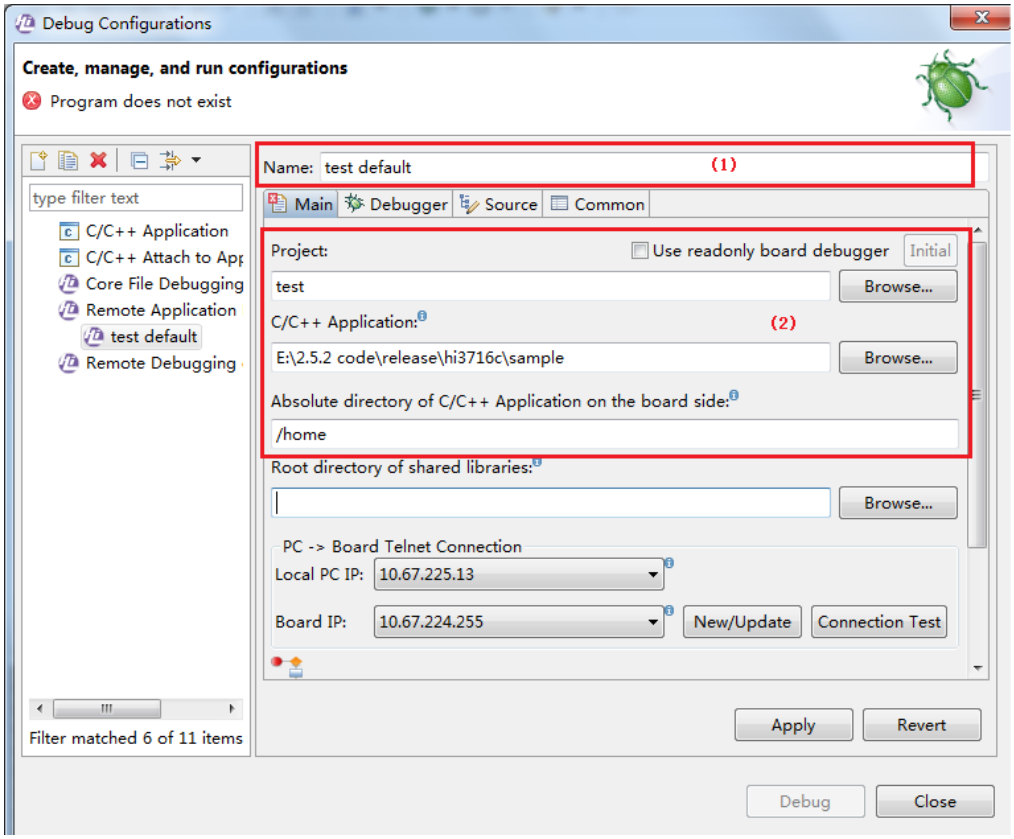


Figure 3-29 Opening the Debug Configurations dialog box by using the toolbar



Step 2 Double-click **Remote Application Debugging** in the displayed **Debug Configurations** dialog box. The HiWorkbench automatically generates and names a configuration. You can change the configuration name in area (1) shown in Figure 3-30. Select the project to be debugged, select a binary file in the project to be debugged, and enter the absolute path of the program on the board. The HiWorkbench automatically downloads the binary file to the path (**/home** by default). See area (2) in Figure 3-30.

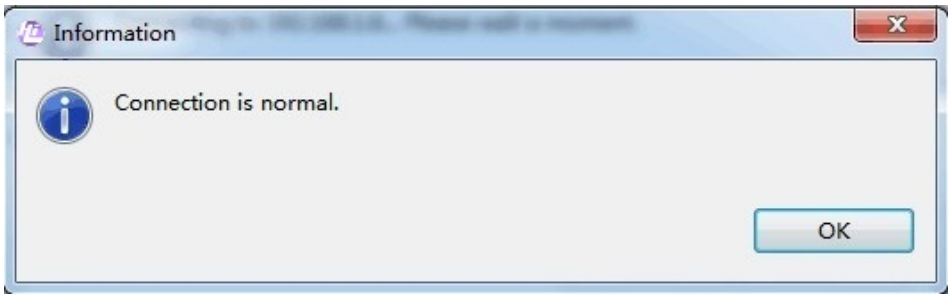
Figure 3-30 Debug Configurations dialog box



Step 3 Select a local IP address from **Local PC IP**.

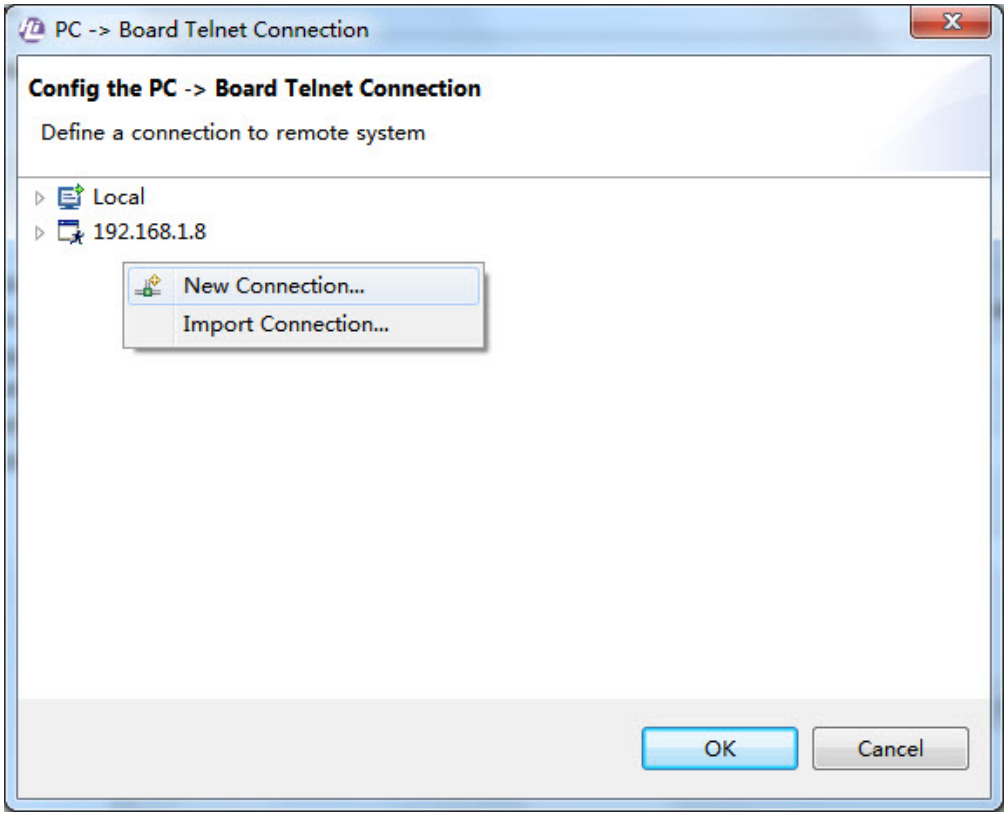
Step 4 Select the IP address for the target board from **Board IP**. Click **Connection Test** to test the connection (optional). After you enter the user ID and password, the dialog box shown in Figure 3-31 is displayed in normal cases. If there is no required IP address in the **Board IP** drop-down list, set a new IP address as follows:

Figure 3-31 Information indicating that the connection is normal



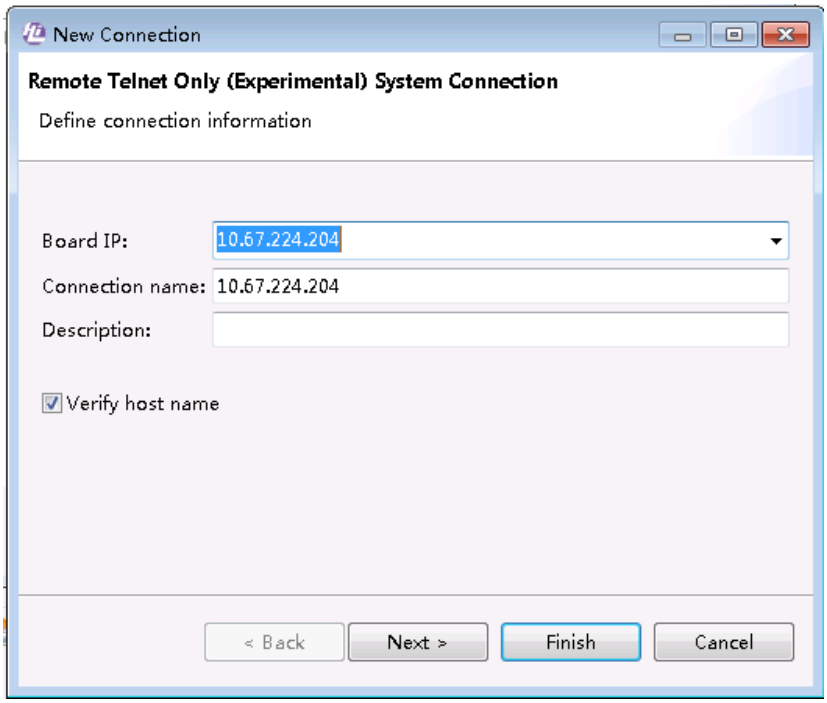
1. Click **New/Update**, right-click the blank area in the displayed dialog box, and choose **New Connection**, as shown in Figure 3-32.

Figure 3-32 Creating a connection



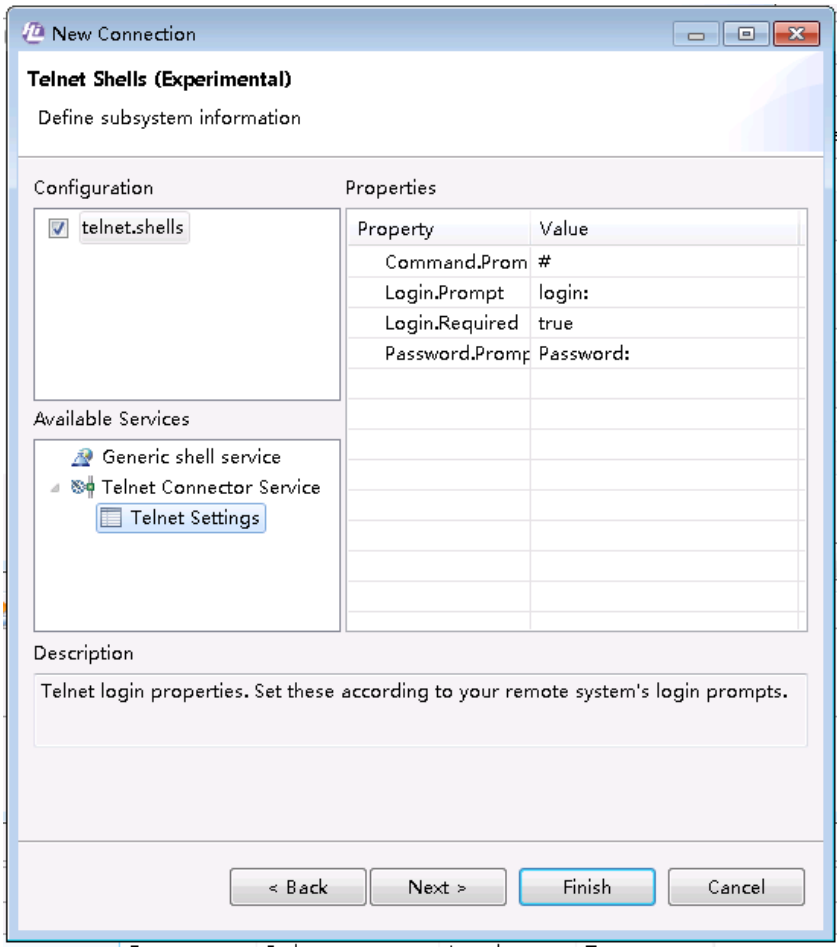
2. Enter the board IP address in the displayed dialog box (ensure that the board IP address has been configured), and click **Finish**, as shown in Figure 3-33. You can also click **Next**.

Figure 3-33 Creating/Updating a connection



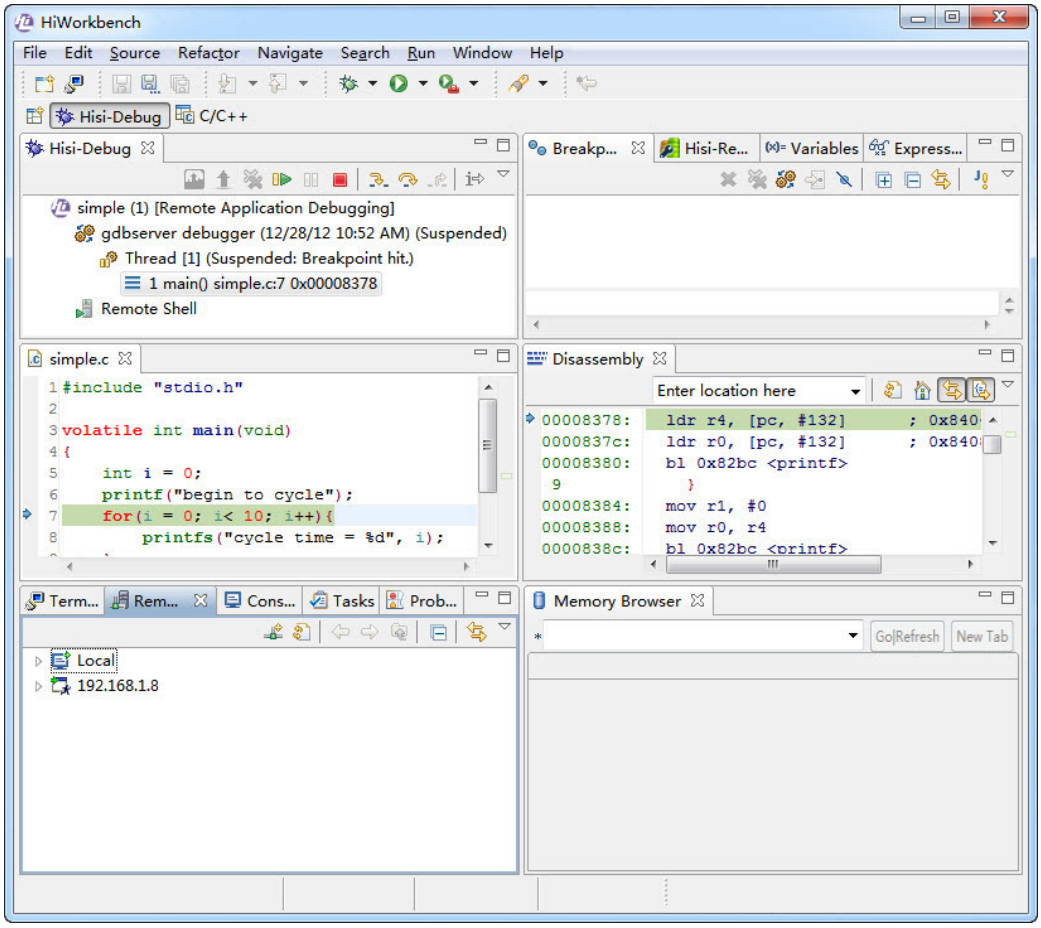
3. (Optional) Set the connection properties. Some boards require that the correct account and password be entered for the telnet connection. The command wildcard character for some boards is #, but that for other boards is not #. This may cause telnet connection failures. These parameters can be configured in the **New Connection** dialog box. Select **Telnet Settings**, as shown in Figure 3-34.

Figure 3-34 Configuring connection properties



Step 5 Click **Debug** to start debugging the application. Figure 3-35 shows the main GUI for debugging.

Figure 3-35 Debugging GUI



For details about debugging configurations of Linux applications, see section 5.3.1 "Linux Application Debugging Configurations."

----End

3.3.2 Debugging a Linux Application in the Read-only File System

If the board file system is read-only, debug the Linux applications as follows:

Step 1 Right-click the project name, and choose **Debug As** > **Debug Configurations**, as shown in Figure 3-36; or choose **Debug Configurations** from the drop-down list of the debug button on the toolbar, as shown in Figure 3-37.

Figure 3-36 Opening the Debug Configurations dialog box by using the shortcut menu

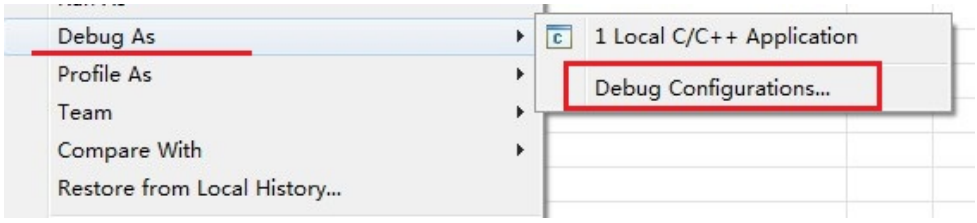
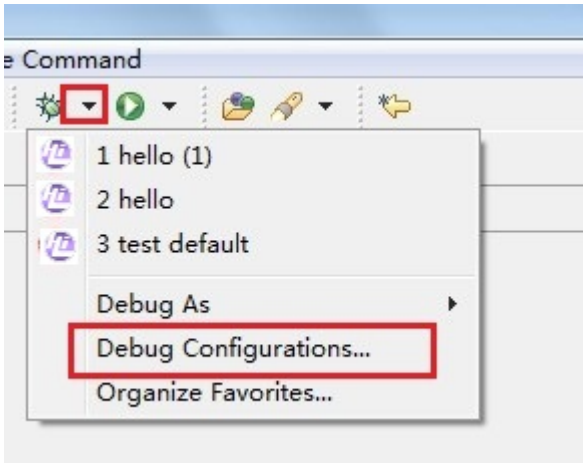
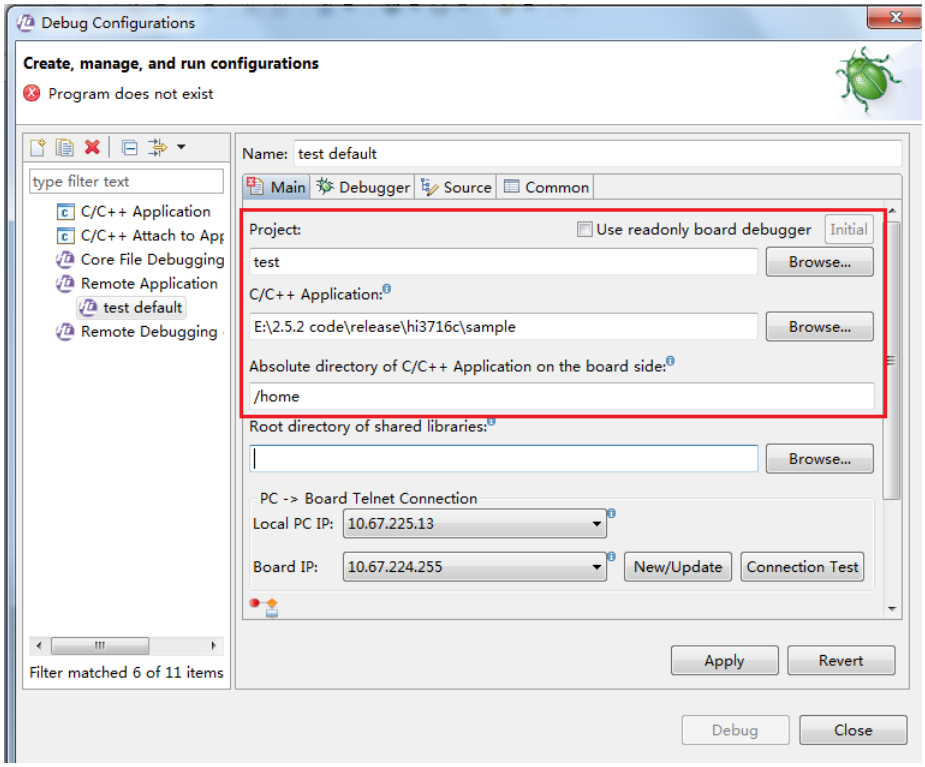


Figure 3-37 Opening the Debug Configurations dialog box by using the toolbar



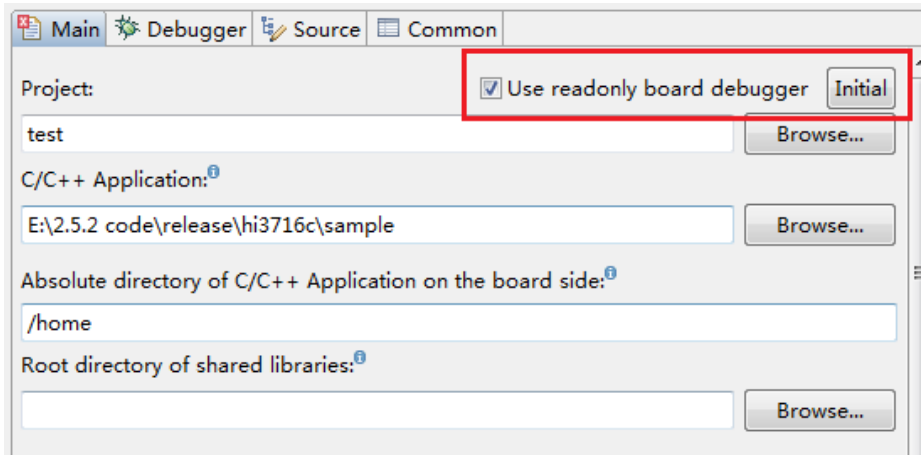
Step 2 Double-click **Remote Application Debugging** in the displayed **Debug Configurations** dialog box. The HiWorkbench automatically generates and names a configuration. Select the project to be debugged, select a binary file in the project to be debugged, and enter the absolute path of the program on the board. The HiWorkbench automatically downloads the binary file to the path (**/home** by default). See Figure 3-38.

Figure 3-38 Debug Configurations dialog box



Step 3 Select **Use readonly board debugger**. The Initial button becomes available, as shown in Figure 3-39.

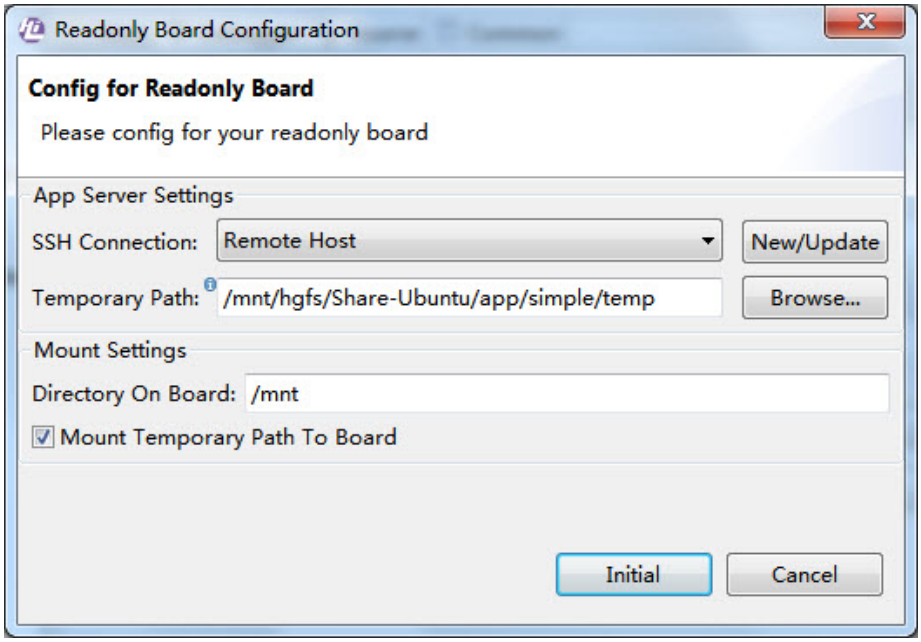
Figure 3-39 Selecting Use readonly board debugger



If the target board uses the read-only file system, the HiWorkbench cannot download the gdbserver for debugging and the binary files to be debugged to the board. Therefore, you need to mount a network file system (NFS) to the target board, and mount files on the server to the target board. Note that the NFS function of the server that provides the mount service must be enabled.

Step 4 Click **Initial**. The **Readonly Board Configuration** dialog box is displayed, as shown in Figure 3-40.

Figure 3-40 Read-only mode debugging configuration

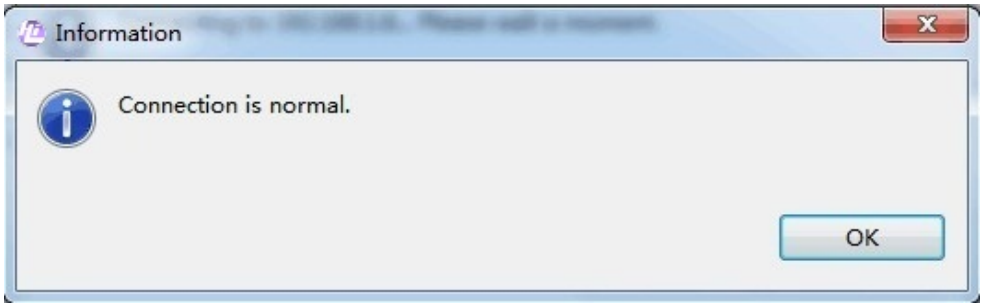


- **SSH Connection**: SSH connection configuration between the target board and the NFS server
- **Temporary Path**: path of files on the NFS server that need to be mounted to the target board
- **Directory On Board**: mount point on the target board
- **Mount Temporary Path To Board**: If **Mount Temporary Path to Board** is selected, the HiWorkbench automatically mounts the path on the server to the target board. If it is not selected, you need to mount the path manually by using the terminal tool.

Step 5 Select the local IP address from **Local PC IP**.

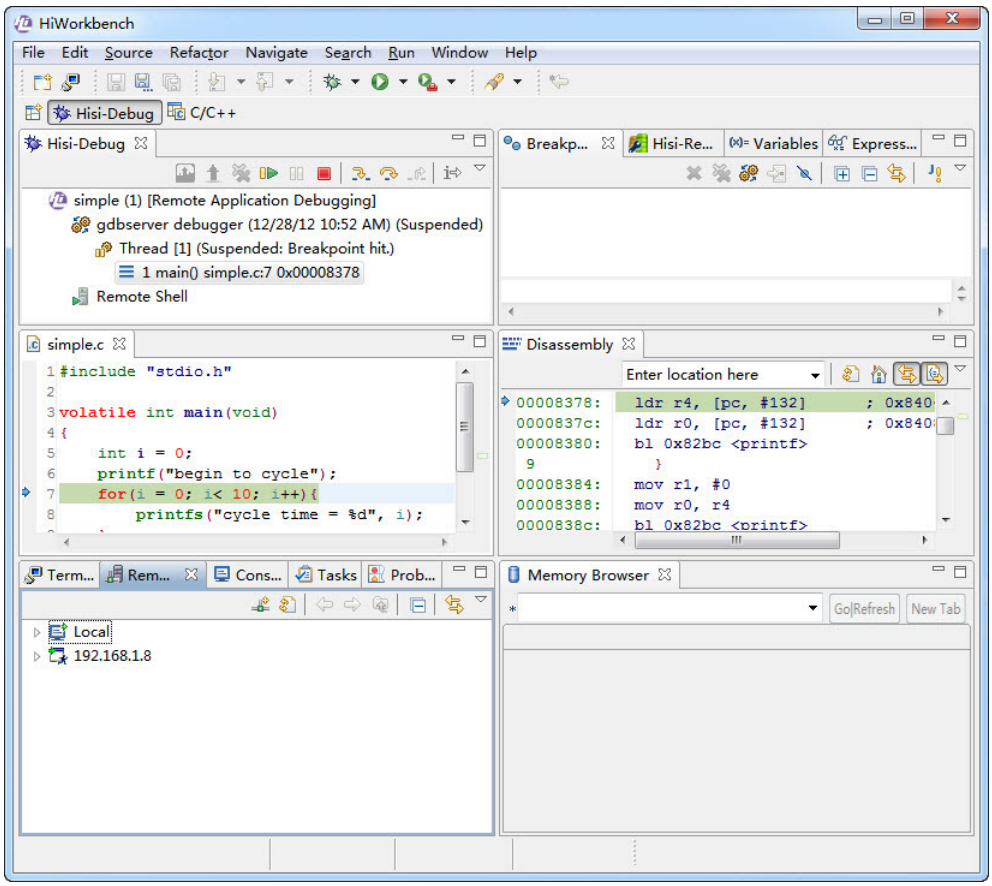
Step 6 Select the IP address for the target board from **Board IP**. Click **Connection Test** to test the connection (optional). After you enter the user ID and password, the dialog box shown in Figure 3-41 is displayed in normal cases. If there is no required IP address in the **Board IP** drop-down list, you need to set a new IP address. For details, see step 4 in section 3.3.1 "Debugging a Linux Application."

Figure 3-41 Information indicating that the connection is normal



Step 7 Click **Debug** to start debugging the application. Figure 3-42 shows the main GUI for debugging.

Figure 3-42 Debugging the application



For details about debugging configurations of Linux applications, see section 5.3.1 "Linux Application Debugging Configurations."

----End

3.3.3 Debugging a Dynamic Linked Library in the Linux Application

To debug a dynamic linked library in the Linux application, perform the following steps:

Step 1 Download the dynamic linked library file to be debugged to the target board. Skip this step if the file is already on the board. Start the TFTP server on the PC end, and download **libtest.so** to the target board by using the HyperTerminal, as shown in Figure 3-43.

Figure 3-43 Downloading the dynamic linked library to the target board

```
# pwd
/usr/lib
# tftp -r libtest.so 192.168.1.100 -g
libtest.so          100% |*******************************|  6144  --:--:-- ETA
```

Step 2 Prepare the dynamic linked library to be debugged and the dynamic loader library file **ld-linux.so** on the PC. See Figure 3-44 and Figure 3-45. Note that the relative root path of the dynamic linked library on the PC must be the same as the absolute path on the board. The **root** directory will be used in subsequent steps.

Figure 3-44 Preparing the dynamic linked library to be debugged on the PC

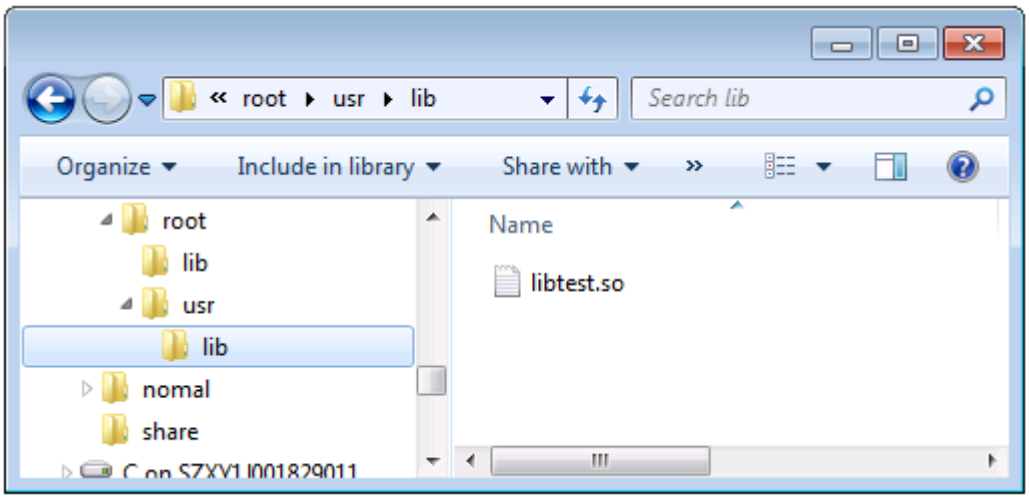
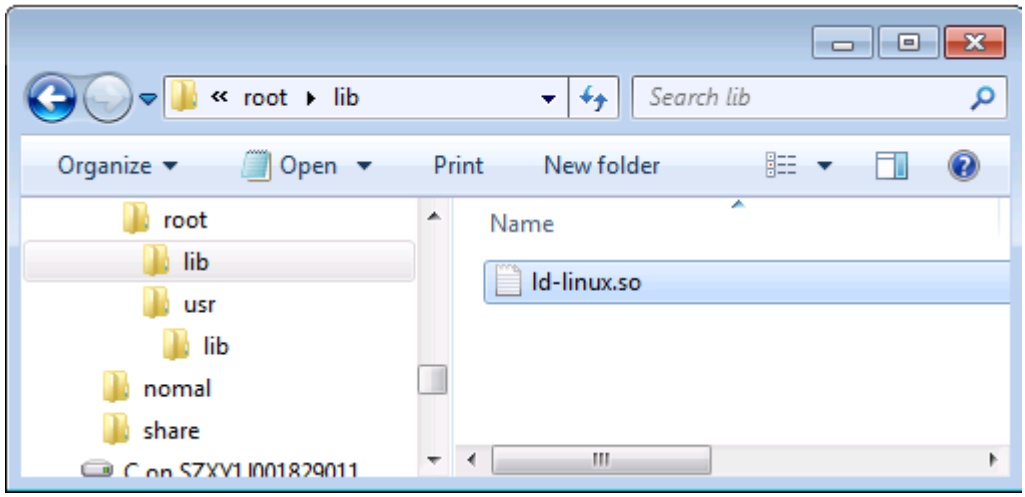
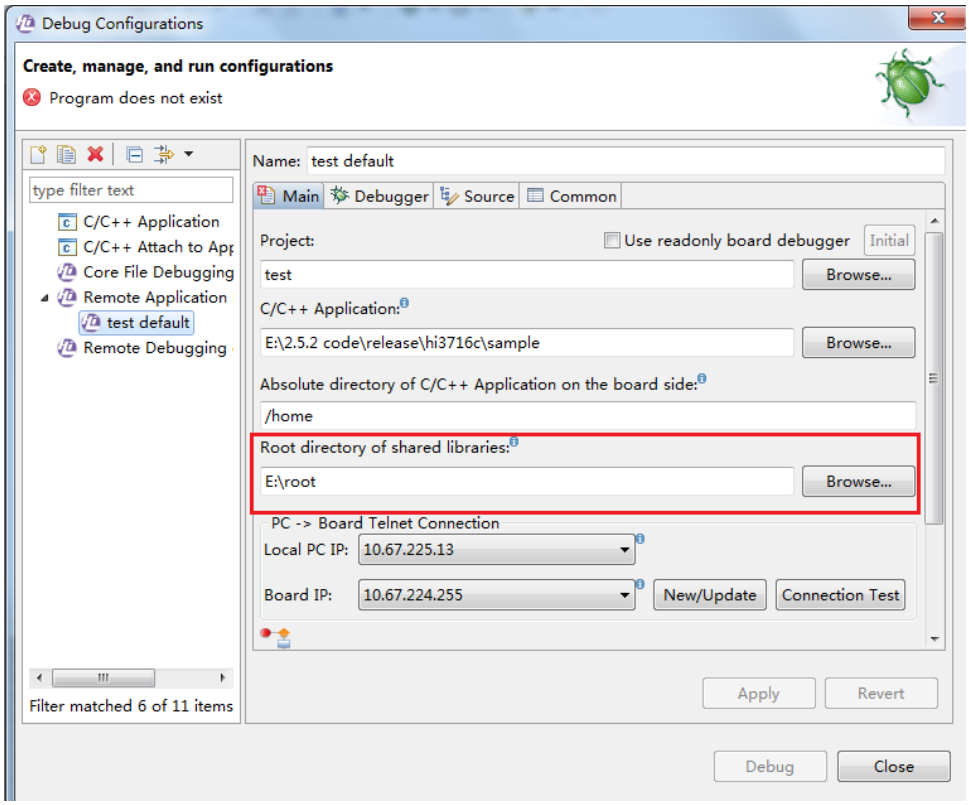


Figure 3-45 Dynamic loader library file



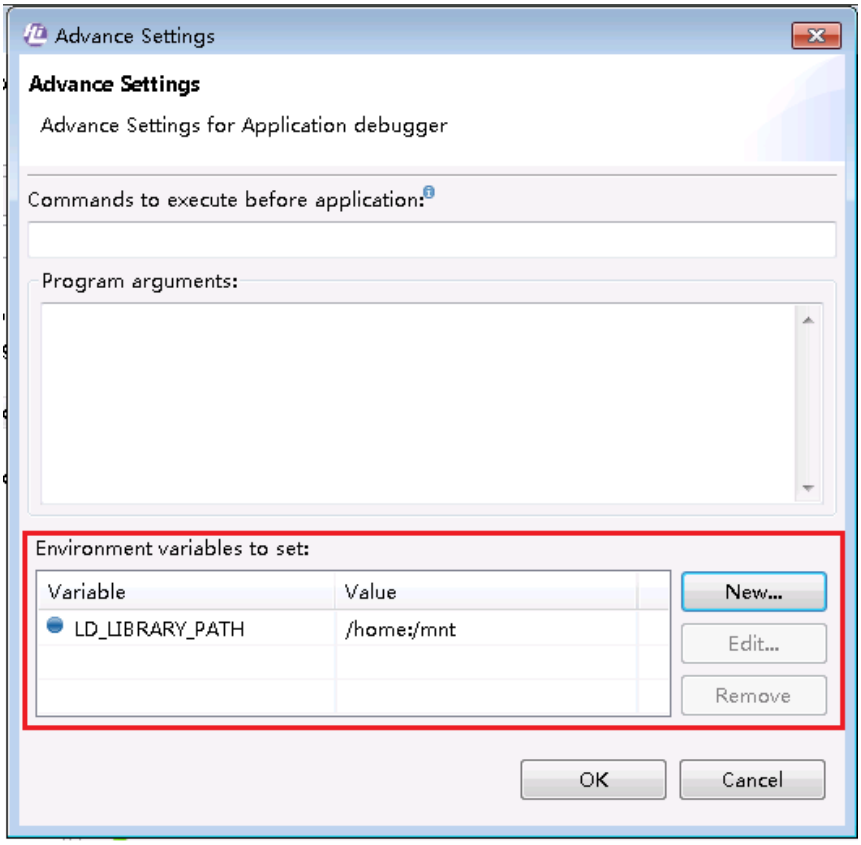
Step 3 Set parameters on the **Debug Configurations** page. For details, see section 3.3.1 "Debugging a Linux Application." Note that you need to set the root directory for dynamic libraries on the PC, as shown in Figure 3-46. The configured path is equivalent to the root directory for the file system on the board.

Figure 3-46 Setting the root directory for dynamic linked libraries on the PC



Step 4 (Optional) Set the **LD_LIBRARY_PATH** environment variable. This environment variable specifies the paths (except the system default paths **/lib** and **/usr/lib**) for searching for dynamic libraries when the program is being loaded. If the dynamic linked libraries are already in **/lib** or **/usr/lib**, skip this step. Click **Advance** on the **Main** tab page, click **New** in the displayed dialog box, and add the required environment variables, as shown in Figure 3-47. Separate the paths by using colons (:).

Figure 3-47 Setting environment variables



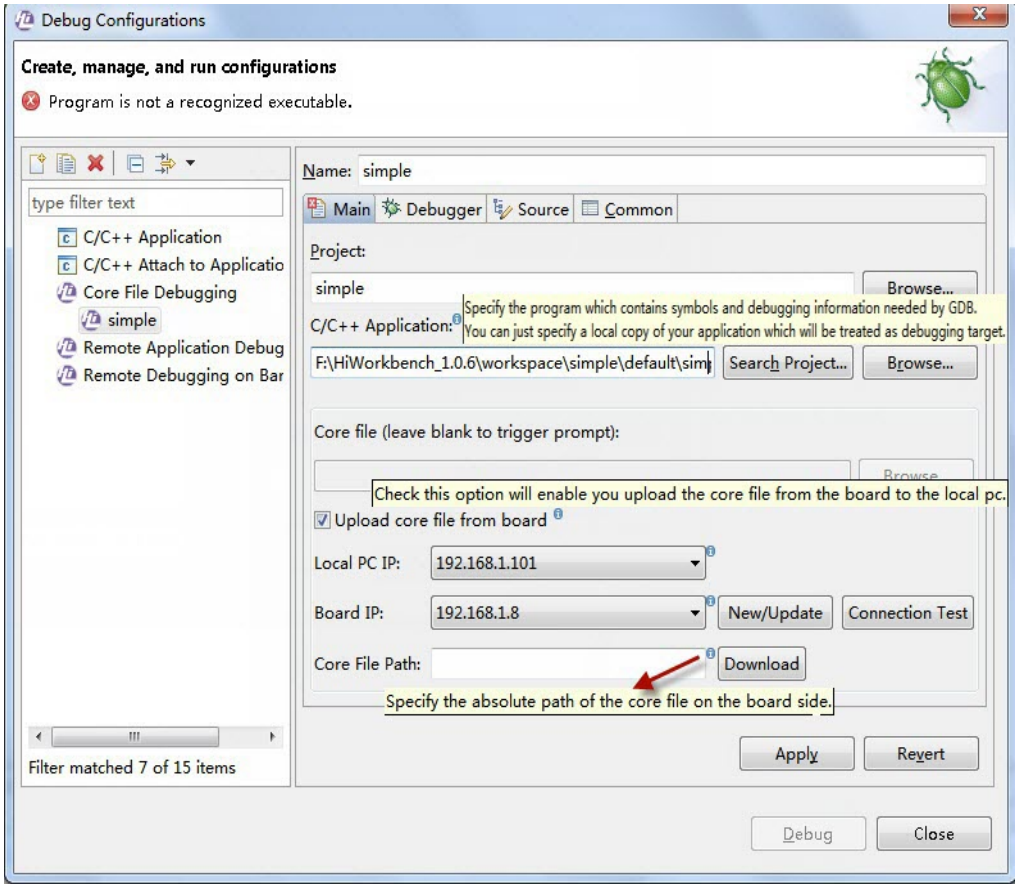
----End

3.3.4 Debugging the Core File

To debug the core file, perform the following steps:

Step 1 Right-click the project name, and choose **Debug As** > **Debug Configurations**, or choose **Debug Configurations** from the drop-down list of the debug button on the toolbar. Double-click **Core File Debugging** in the **Debug Configurations** dialog box. Then the HiWorkbench automatically generates a new configuration with the name of the project, as shown in Figure 3-48.

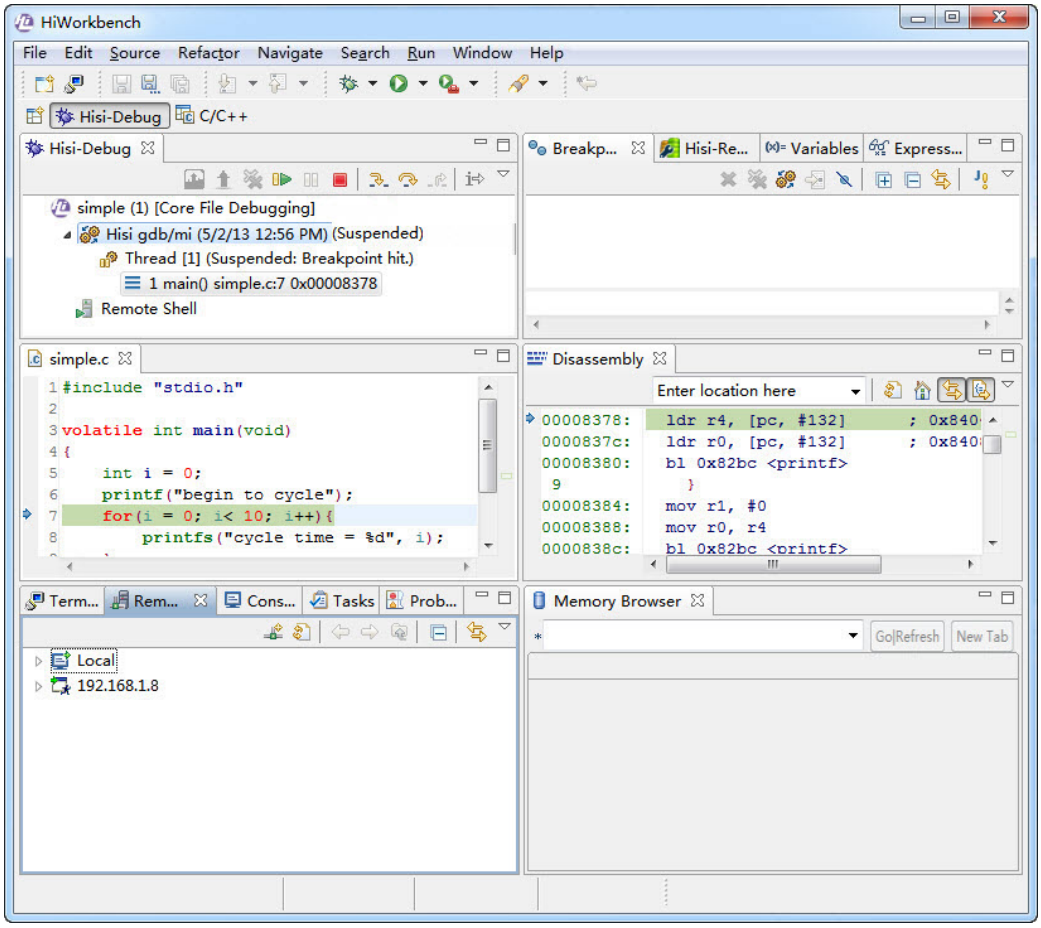
Figure 3-48 Debug Configurations dialog box



Step 2 Set **Project** and **C/C++ Application**, and select a local core file; or select **Upload core file from board**, and download the core file from the target board to the local. If **Upload core file from board** is selected, set **Local PC IP**, **Board IP**, and **Core File Path**, and click **Download** to obtain the core file.

Step 3 Click **Debug** to start debugging the core file. Figure 3-49 shows the main GUI for debugging.

Figure 3-49 Debugging the core file



For details about core file debugging configurations, see section 5.3.3 "Core File Debugging Configurations."

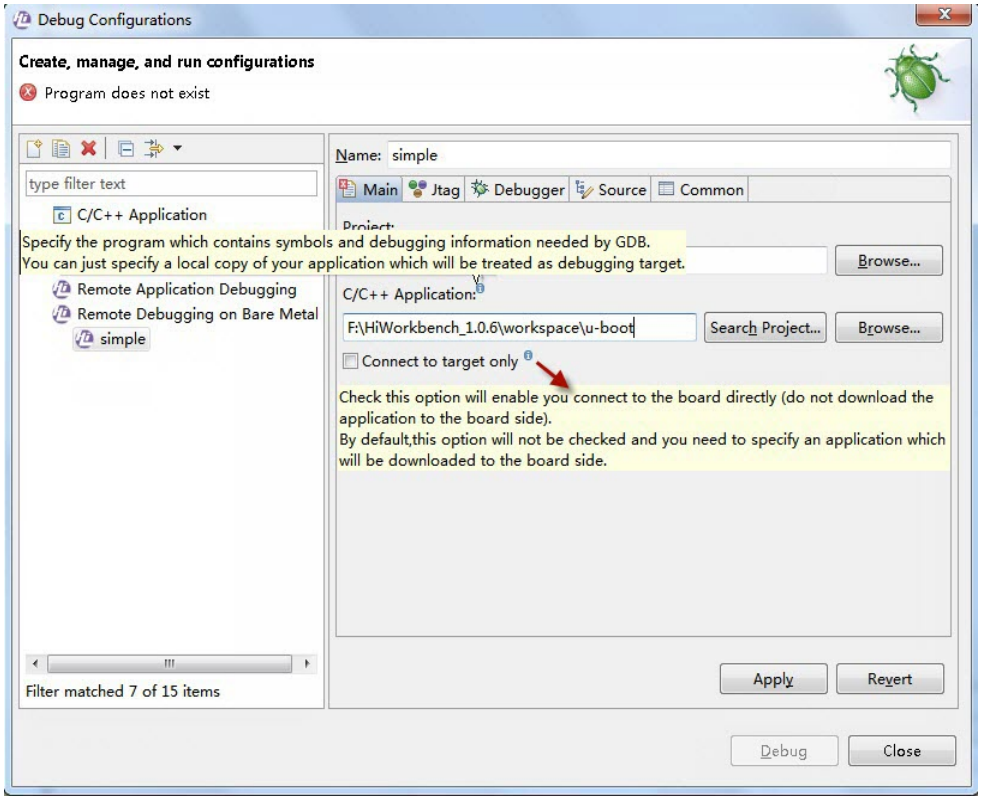
----End

3.3.5 Debugging the Bare Board Program

To debug the bare board program, perform the following steps:

Step 1 Right-click the project name, and choose **Debug As** > **Debug Configurations**, or choose **Debug Configurations** from the drop-down list of the debug button on the toolbar. Double-click **Remote Debugging on Bare Metal** in the **Debug Configurations** dialog box. Then the HiWorkbench automatically generates a new configuration with the name of the project, as shown in Figure 3-50.

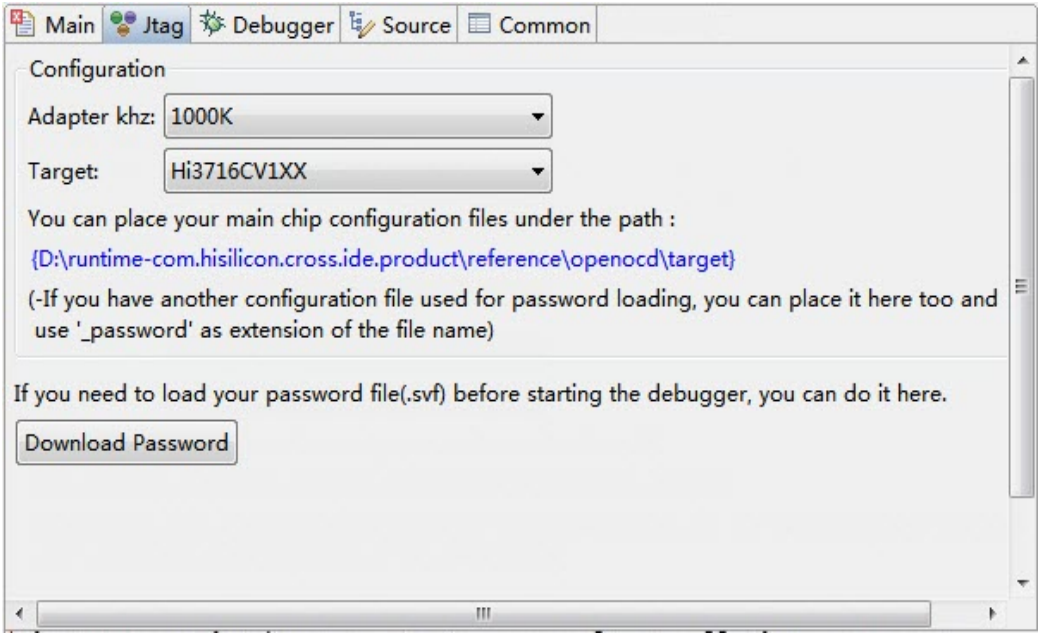
Figure 3-50 Debug Configurations dialog box



Step 2 Select the project to be debugged and the ELF file after compilation, or select **Connect to target only** to connect to the board.

Step 3 Click the **Jtag** tab, and set the JTAG connection frequency and chip model of the target board. See Figure 3-51.

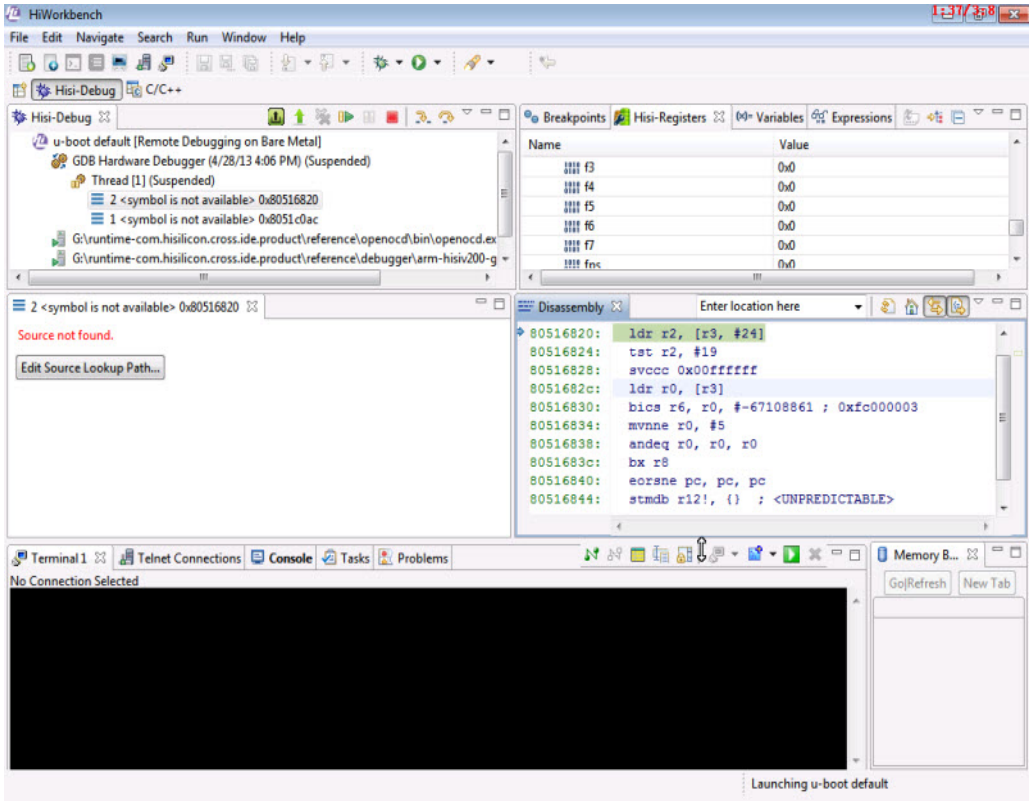
Figure 3-51 Jtag tab page



Step 4 (Optional) Click **Download Password**, and select the serial vector format (SVF) key file to be downloaded.

Step 5 Click **Debug** to start debugging the bare board program. Figure 3-52 shows the main GUI for debugging.

Figure 3-52 Debugging the bare board program



Step 6 (Optional) Initialize the target board DDR.

For details about bare board debugging configurations, see section 5.3.2 "Bare Board Program Debugging Configurations."

----End

3.4 Managing Connections

This section describes how to manage connections.

3.4.1 Managing Board Connections

The **Telnet Connections** tab page is used to manage board-end network connections. Click the icon in the red rectangle in Figure 3-53 to open the **Telnet Connections** tab page. You can then create, update, or delete connections. You can also use the shortcut menu by right-clicking the items on the tab page. See Figure 3-54.

Figure 3-53 Telnet Connections tab page

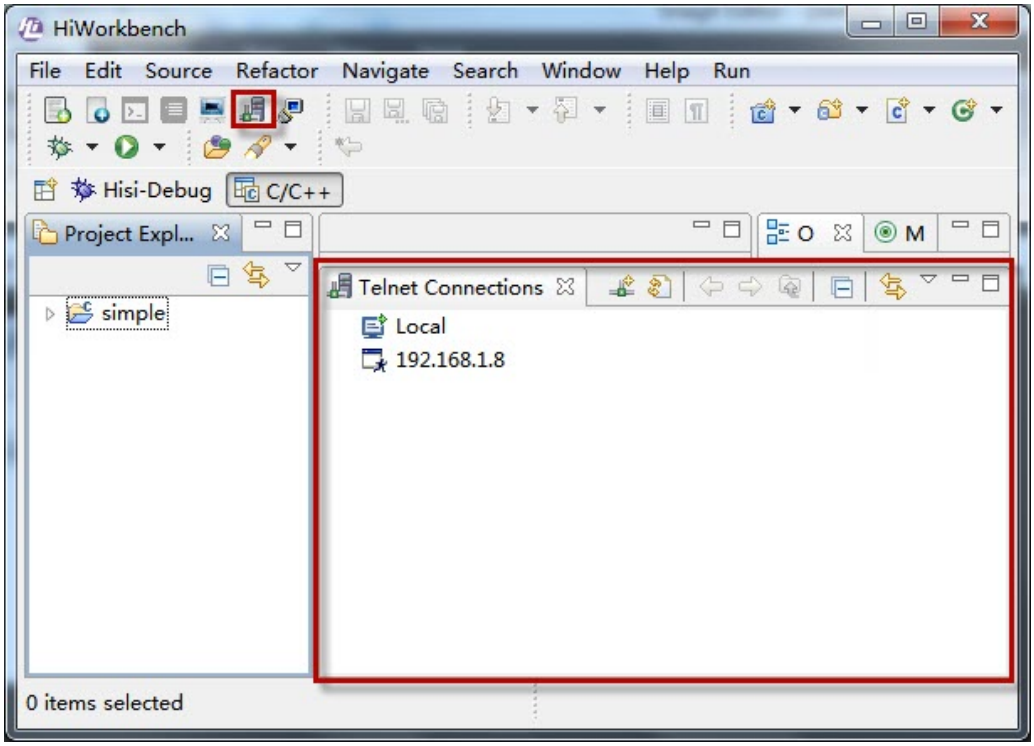
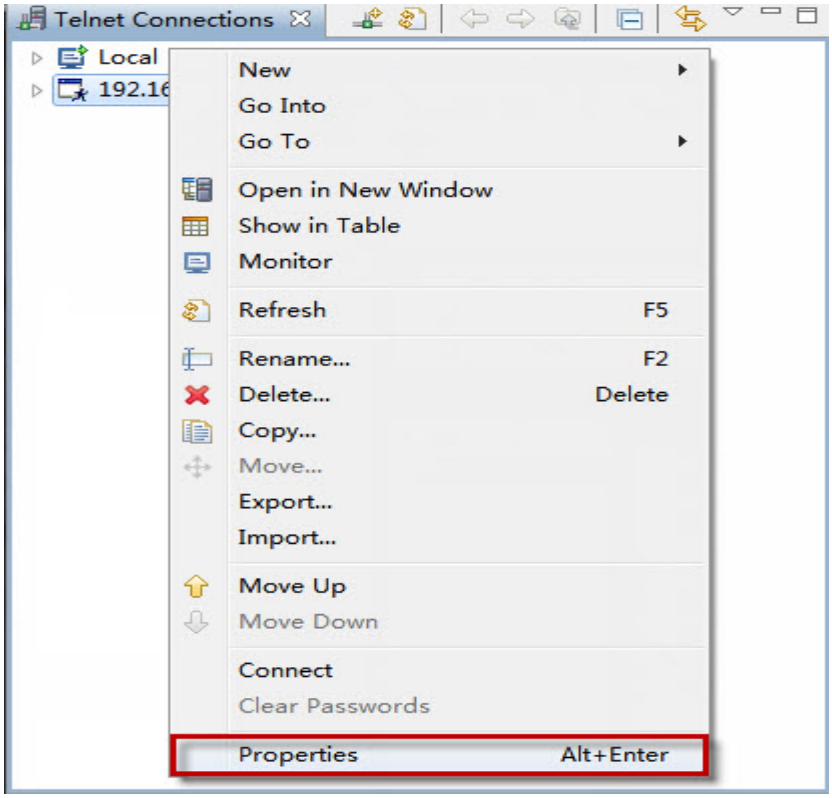


Figure 3-54 Shortcut menu



Choose **Properties** from the shortcut menu. A dialog box is displayed, allowing you to update the connection. Set the following parameters based on the board output information, as shown in Figure 3-55.

- **Command.Prompt**: start prompt of the command-line interface (CLI) after login over the Telnet. It is # by default.
- **Login.Prompt**: character string for reminding when you enter the user ID during the telnet login. It is **login:** by default.
- **Login.Required**: whether the user ID and password are required during telnet login. If it is **true**, a dialog box is displayed upon the first connection, asking you to enter the user ID and password, as shown in Figure 3-56.
- **Password.Prompt**: password prompt during the telnet login

Figure 3-55 Connection properties

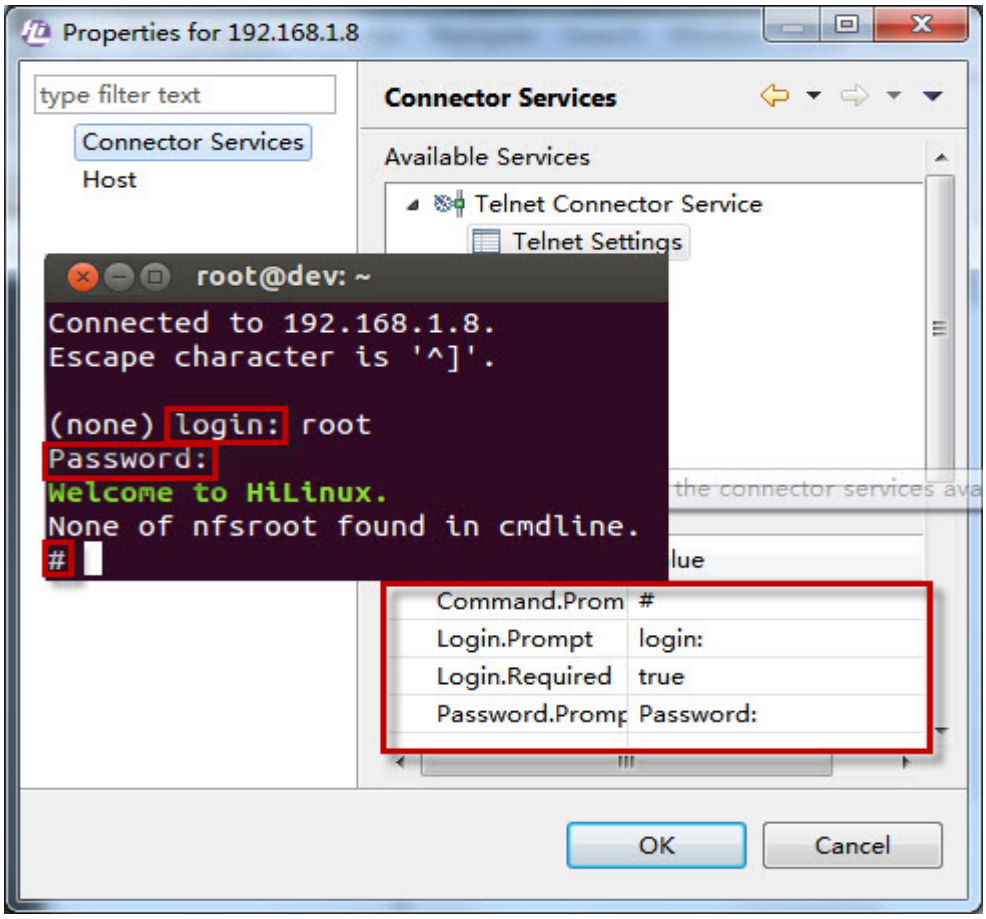
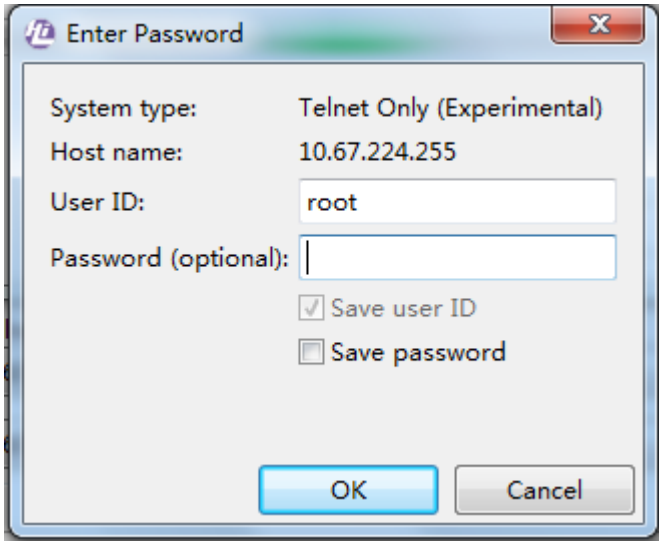


Figure 3-56 Entering the user ID and password

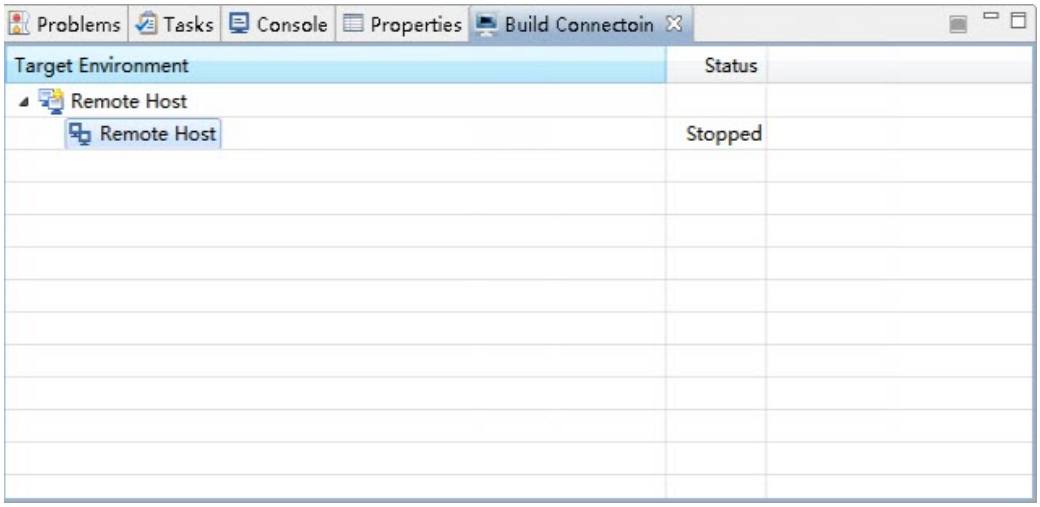


3.4.2 Managing Compilation Connections

This section describes how to manage SSH compilation connections.

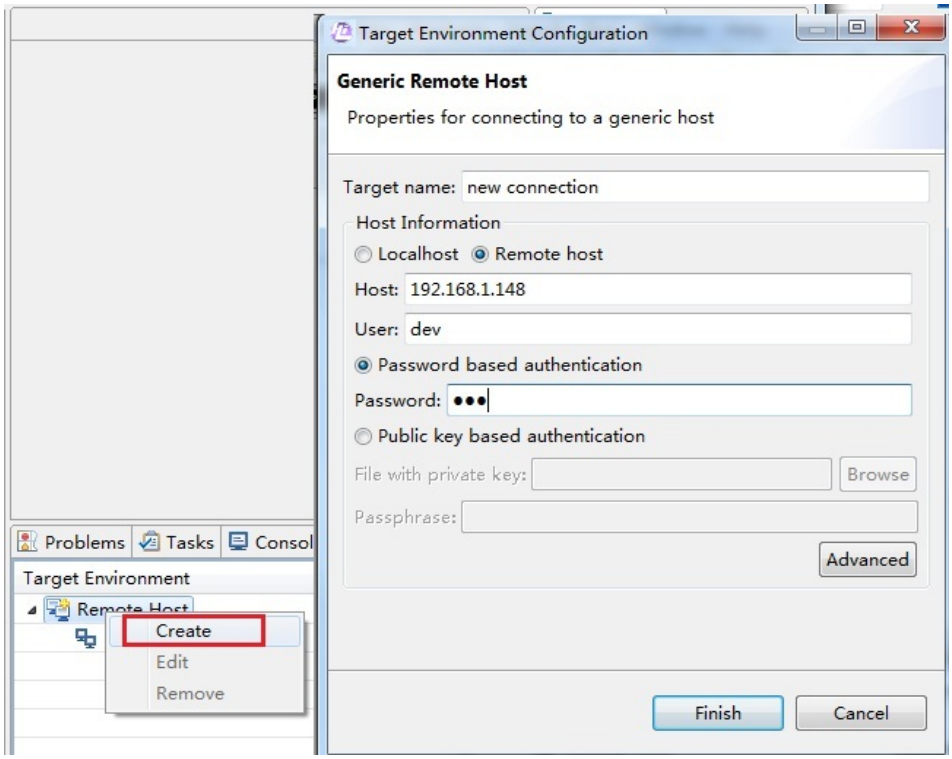
The **Build Connection** tab page is used to manage all connections during the process of creating remote applications. You can create, edit, and delete connections on this tab page. You can also know about the status of each connection. See Figure 3-57.

Figure 3-57 Build Connection tab page



Right-click an item in the **Target Environment** column, and choose **Create** from the shortcut menu. The dialog box shown in Figure 3-58 is displayed. Enter the connection name, select **Remote host**, and specify the host IP address, user ID, and password. Click **Finish**. The method for editing an existing connection is similar.

Figure 3-58 Creating a connection



3.5 Automatically Creating the makefile File

To configure the compilation options and create the **makefile** file, perform the following steps (supported by only empty projects):

Step 1 Right-click the project name, and choose **Makefile Configuration** from the shortcut menu, as shown in Figure 3-59; or click  . The dialog box shown in Figure 3-60 is displayed.

Figure 3-59 Selecting Makefile Configuration from the shortcut menu

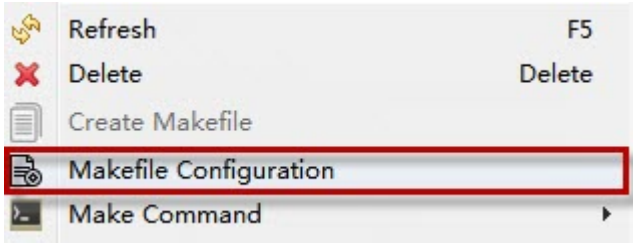
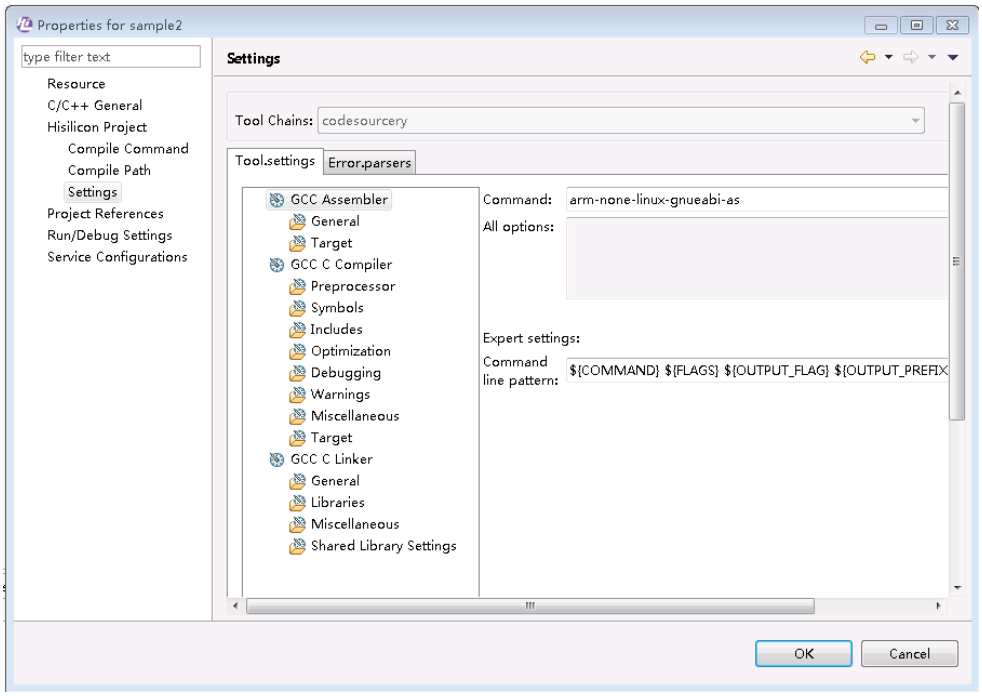
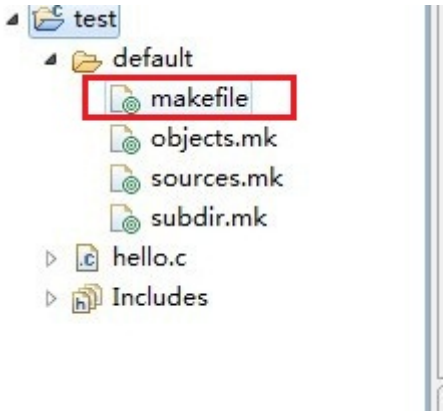


Figure 3-60 Setting compilation options



Step 2 Set the parameters on the **Tool.settings** tab page, and click **OK**. A **makefile** file is created in the **default** directory. See Figure 3-61.

Figure 3-61 Generating the makefile file



You can change the project tool chain in the **Tool Chains** drop-down list.

----End

4 Important Concepts

4.1 HiSilicon Projects

The HiWorkbench supports the following project types (excluding the CDT C/C++ projects):

- **Project With Existing Code**

  The source code and **makefile** file already exist in projects of this type. Therefore, the HiWorkbench does not generate the **makefile** file automatically, and you need to add and edit the file manually.

  Because some directories may have been created in the root directories of projects of this type, the HiWorkbench uses some optimization policies such as file filtering to accelerate project building.

- **Empty Project**

  There is no file (including the **makefile** file) in the root directory of empty projects. Therefore, you can use the HiWorkbench to automatically generate the **makefile** file required for compilation. However, you need to configure some necessary parameters. Note that the HiWorkbench does not apply optimization policies to empty projects.

Keywords:

- **Remote**: The project is created and compiled on a remote host. You need to specify an SSH connection for compiling the remote project.
- **Local**: The project is created and compiled on a local host.
- **ToolChain**: Each tool chain contains the corresponding header file and compilation commands. You need to specify the tool chain for a project when creating it.

4.2 Project Compilation

The HiWorkbench supports two compilation modes:

- Remote compilation

  If the project is on a remote host and the tool chain has been installed on the remote host, remote compilation can be used. In this case, you need to create an SSH connection to the remote host so that the HiWorkbench can log in to the remote host and run compilation commands.

- Local Compilation

If the project is in the local host and the tool chain has been installed on the local host, you can compile the project without any connection.

The following are the GNU compiler collection (GCC) options:

- *Debugging option (***-g***)*

  Debugging information can be generated in the OS inherent format (stabs, COFF, XCOFF, or DWARF).

  The GUN debugger (GDB) can work with the debugging information.

  On most systems that use the stabs format, the **-g** option allows extra debugging information to be used by only the GDB, and the extra information makes the GDB work better.

- Optimization option (**-o**). The GCC allows you to use the **–o** option with the **-g** option. Code optimization may result in surprising results:
  - Some variables you declared may not exist at all.
  - Flow of control may briefly move where you do not expect it.
  - Some statements may not be executed because they compute constant results or their values are already at hand.
  - Some statements may execute in different places because they have been moved out of loops.

  However, it proves that the output can be debugged and optimized. You can optimize the programs that may have bugs properly.

4.3 HiSilicon Debugger

The HiWorkbench supports three types of debuggers:

- Core file debugger

  A core file is a basic file. It contains full status information when an application crashes. The HiWorkbench allows the core file to be loaded by using the GDB and provides a friendly UI to check the program memory and processor registers (including information about program quantity, stack points, and OS and memory management).

- Linux application debugger

  The HiWorkbench debugs the Linux application on the target board by using the GDB and GDB server. During debugging, the GDB server runs on the target board, and the GDB runs on the local host. The GDB and GDB server communicate by using the standard GDB remote serial protocol over the TCP connection.

- Bare board program debugger

  The HiWorkbench debugs the bare board program by using the GDB and OpenOCD (using the remote GDB server protocol to debug remote targets). To use this debugger, a JTAG emulator Hi-ICE is required. The HiWorkbench provides some standard embedded configuration files used by the OpenOCD for HiSilicon chips. It also allows you to add configuration files for your chips.

4.4 Connections

The HiWorkbench supports two types of connections:

- Telnet connection (between a host and a board)

  This type of connection is used for program debugging, for example, when the HiWorkbench needs to send commands or transfer files to the PC or board.

- SSH connection (between a local host and a remote host)

  If you need to compile a project on a remote host, an SSH connection for compilation needs to be established. The HiWorkbench logs in to the remote host and executes compilation commands over the SSH connection.

4.5 Script File Formats

- SVF format

  The SVF is a format indicating that the JTAG test model has been passed. It uses the transmission protocol that debugs sessions with the JTAG. The OpenOCD supports the running of these test files.

  The following is an example of the .svf file.

```
------------------------------------------------------------------------
------
! attention TRSTN not controlled by JtagStudio, you should reset TRSTN manually
FREQUENCY 1000000 HZ;
STATE RESET;
STATE IDLE;
!RUNTEST IDLE 1000 TCK ;

!!!!!!!!!!!!!!!!!!!!
! read_otp 0
!!!!!!!!!!!!!!!!!!!!

! jtag_cs is high
SIR 6 TDI (30) SMASK (3F) TDO (01) MASK (00);
SDR 84 TDI (000000000004000000001) SMASK (ffffffffffffffffffff) TDO
(ffffffffffffffffffff) Mask(00000000000000000000);

! jtag_cs is low
SIR 6 TDI (30) SMASK (3F) TDO (01) MASK (00);
SDR 84 TDI (000000000004000000000) SMASK (ffffffffffffffffffff) TDO
(ffffffffffffffffffff) Mask(00000000000000000000);

!get otp value
SIR 6 TDI (30) SMASK (3F) TDO (01) MASK (00);
SDR 97 TDI (000000000000000000000000) SMASK (000000000000000000000000) TDO
```

```
(00000000000000000000ffff) Mask(00000000000000000000ffff);

!!!!!!!!!!!!!!!!!!!!
! read_otp 1
!!!!!!!!!!!!!!!!!!!!
------------------------------------------------------------------------
------
```

- Common command format

  Commands in the script can be directly executed by the OpenOCD. For details about more commands, see the *OpenOCD User's Guide*. A row is considered as a comment line if it starts with // or #.

  The following is an example of the command file.

```
------------------------------------------------------------------------
------
// config DDRC IO setting
// ddr3=0x0, rtt=0x2
mww 0x10100570 0x427
mww 0x10100574 0x427

// dll reset
mww 0x10100430 0x52
mww 0x10100434 0x52

// waiting...
sleep 0x1

// dll reset removal
mww 0x10100430 0x53
mww 0x10100434 0x53

// waiting for dll lock
sleep 0x1

// disable auto refresh
mww 0x10100058 0x82208000

// waiting 1 ms...
sleep 0x1

// exit self-refresh, and enter normal mode
mww 0x10100004 0x0

// wait until in_sr LOW
```

```
sleep 0x1

// config DDR MRS01
// WR=0x2, CL=0x5, BL=0x1
mww 0x10100014 0x60253
// Rtt=0x1, ODIC=0x1
mww 0x10100018 0x0
------------------------------------------------------------------------
------
```

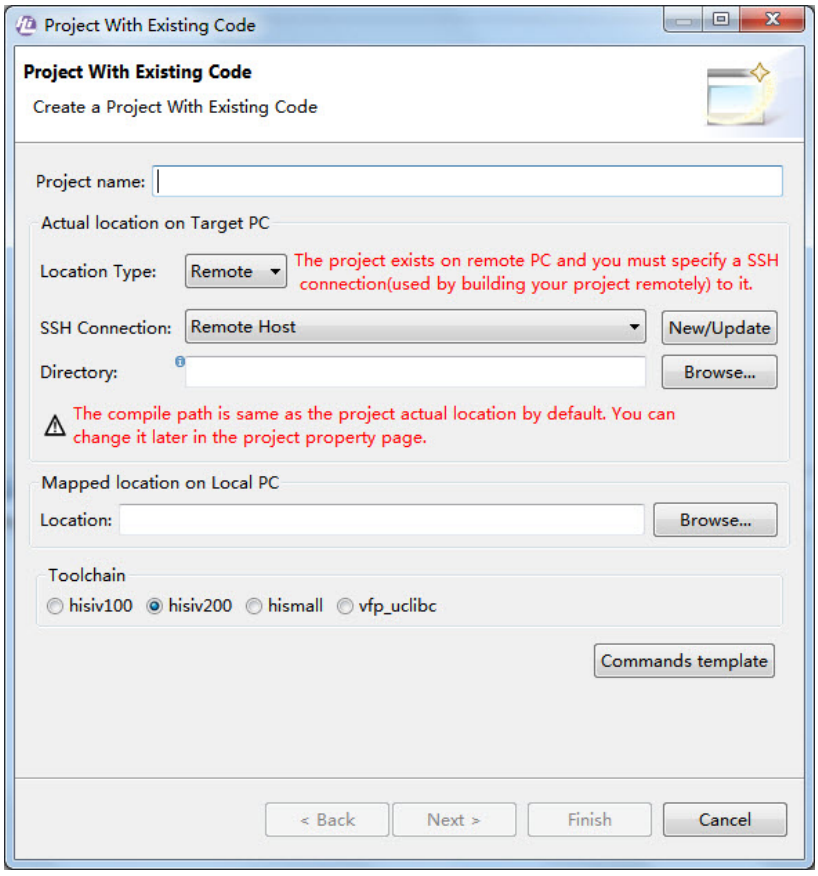
5 UI Components

This chapter describes the UI components of the HiWorkbench.

5.1 Creating a Project

Figure 5-1 shows the main page for creating a project.

Figure 5-1 Main page for creating a project.



- **Project name**: name of the project to be created

- **Location Type**: location for storing the project and source code (local host or remote server). Select **Remote** if the project needs to be compiled on a remote server, and select **Local** if the project needs to be compiled on the local PC.
- **SSH Connection**: This option needs to be configured if **Location Type** is **Remote**. The configured parameters are used to create a connection to the remote server.
- **Directory**: This option needs to be configured if **Location Type** is **Remote**. You need to specify a directory on the remote server. During compilation of the project, the compilation commands are executed under this directory.
- **Mapped location on Local PC**: mapped path on the local host for the path of the project on the remote server
- **Toolchain**: tool chain used for compiling the project
- **Commands template**: compilation command template. See Figure 5-2.

Figure 5-2 Command template

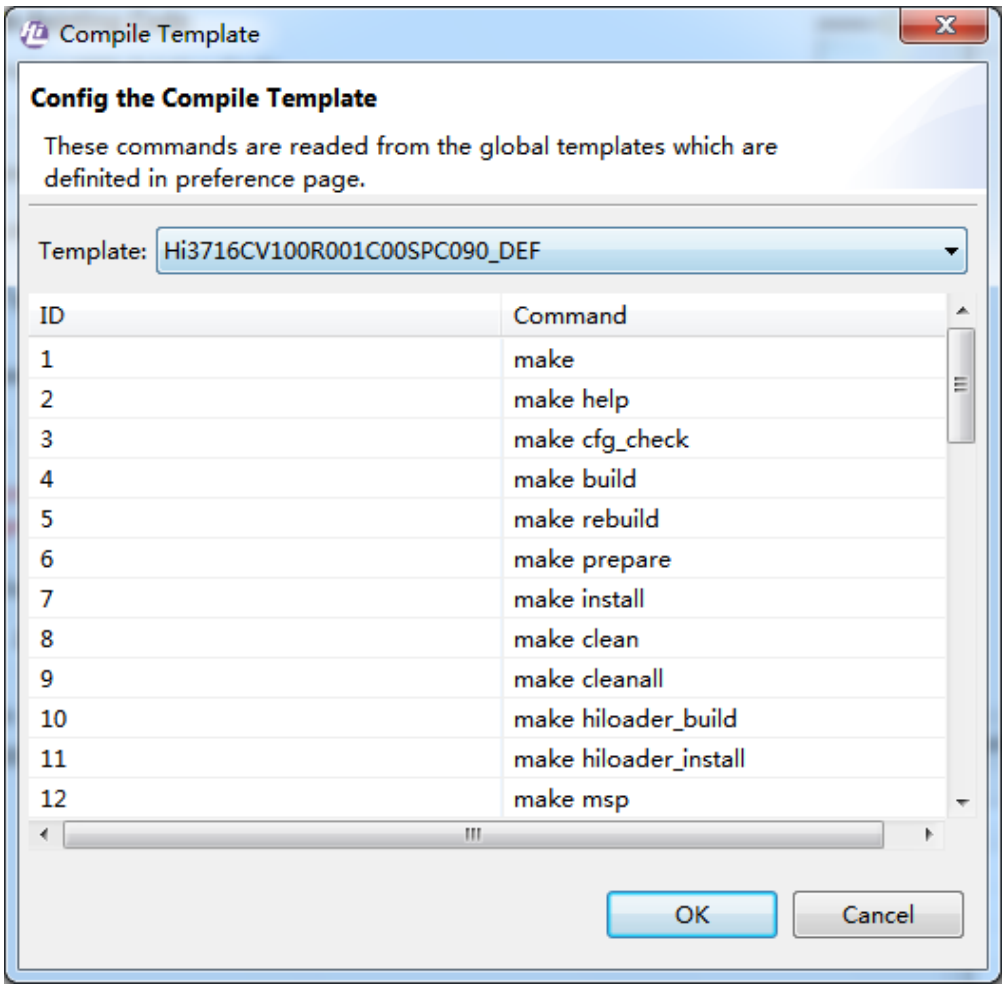
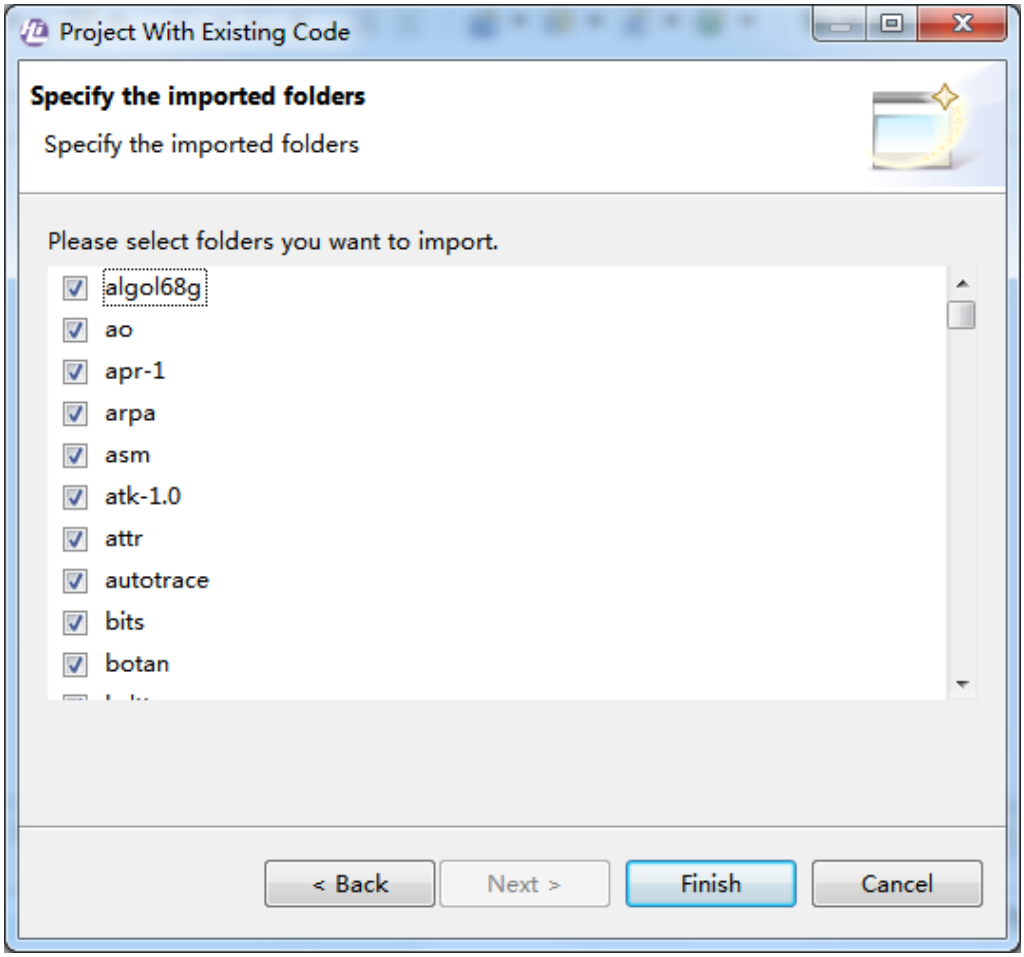


Figure 5-3 shows the GUI for importing directories. You can specify the top directories in the specified project path to be imported, and only specified directories are visible in the project. All top directories are selected by default.

Figure 5-3 GUI for importing directories

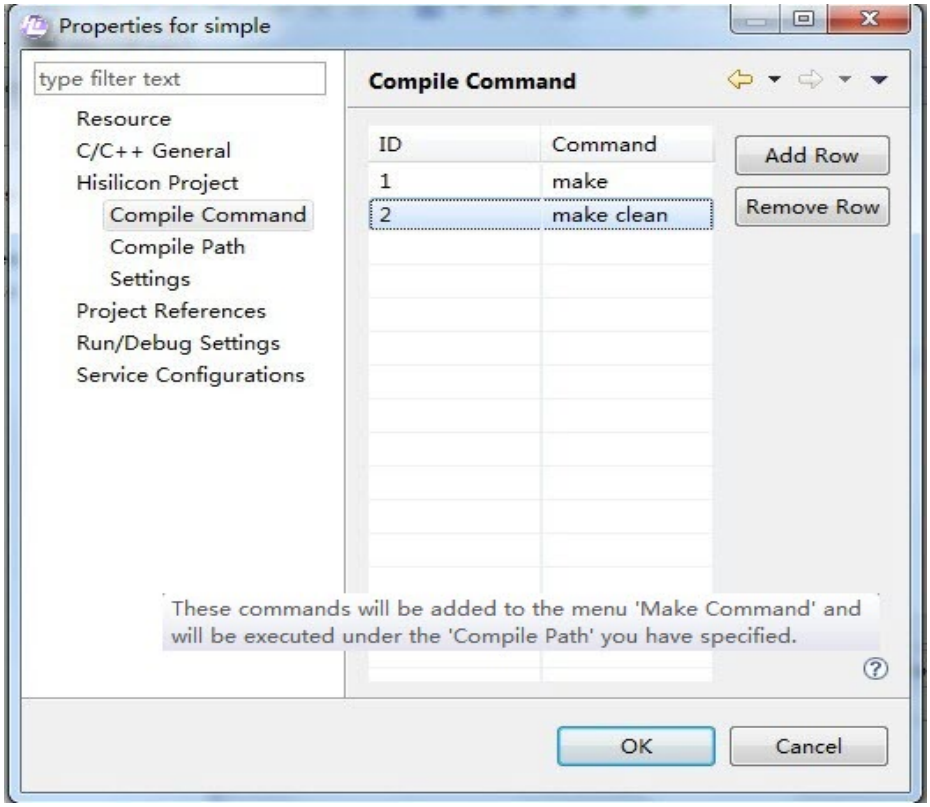


5.2 Updating Compilation Commands

5.2.1 Project Compilation Commands

Right-click a project, and choose **Properties** form the shortcut menu. Choose **Hisilicon Project** > **Compile Command**, as shown in Figure 5-4. Click **Add Row** to add a command, or click **Remove Row** to remove a selected command.

Figure 5-4 Compile Command



The commands are added to the **Make Command** menu, which can be viewed on the menu bar or the shortcut menu.

If a template is selected when a project is created, all commands are displayed in the **Make Command** menu of the menu bar and the shortcut menu, as shown in Figure 5-5 and Figure 5-6.

Figure 5-5 Make Command menu on the menu bar

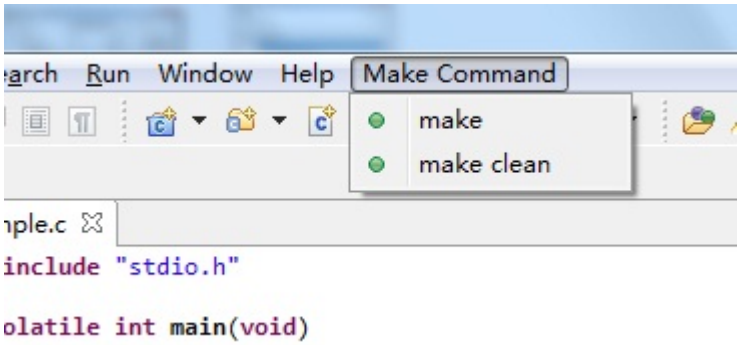
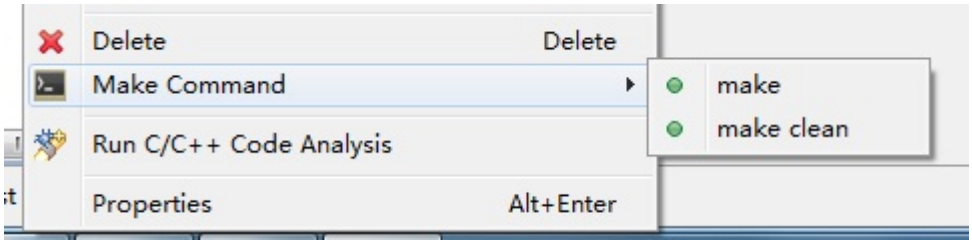


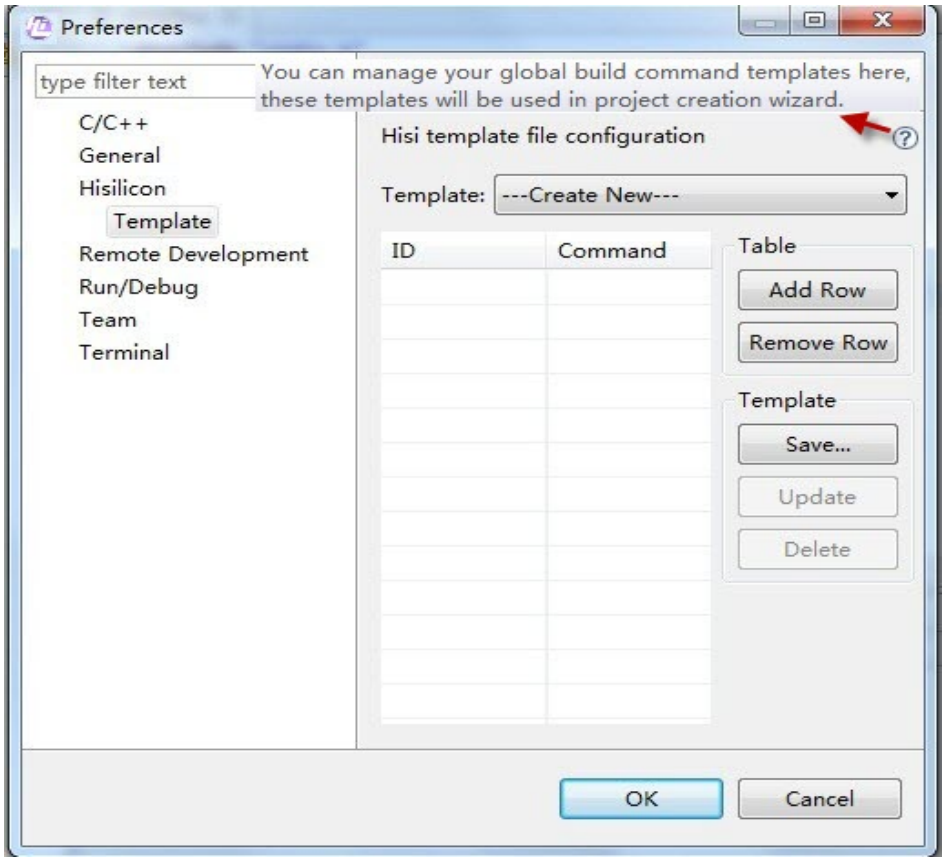
Figure 5-6 Make Command menu in the shortcut menu



5.2.2 Global Compilation Commands

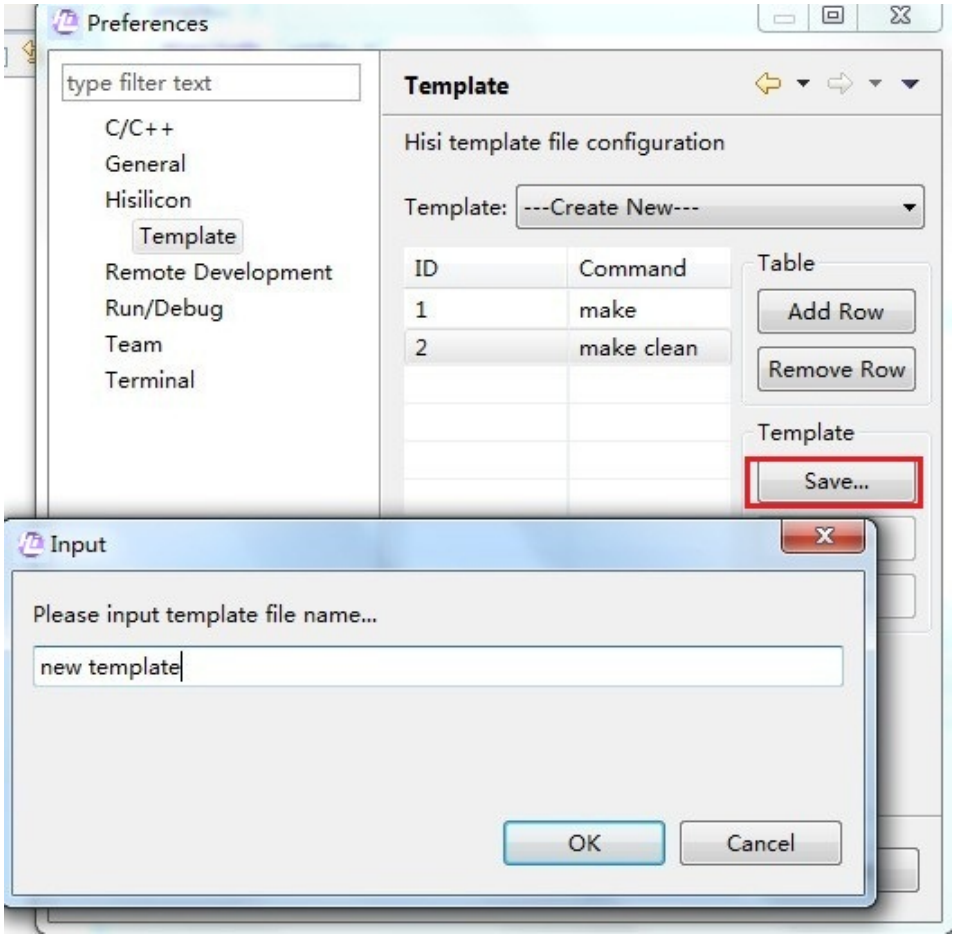
Choose **Windows** > **Preferences**. The **Preferences** dialog box is displayed. Choose **Hisilicon** > **Template**. See Figure 5-7.

Figure 5-7 Template



You can create a template or modify an existing template by selecting the template from the **Template** drop-down list. If you select **Create New**, add commands in the table below by clicking **Add Row**, click **Save**, and enter a new template name. You can also remove commands by clicking **Remove Row**. See Figure 5-8.

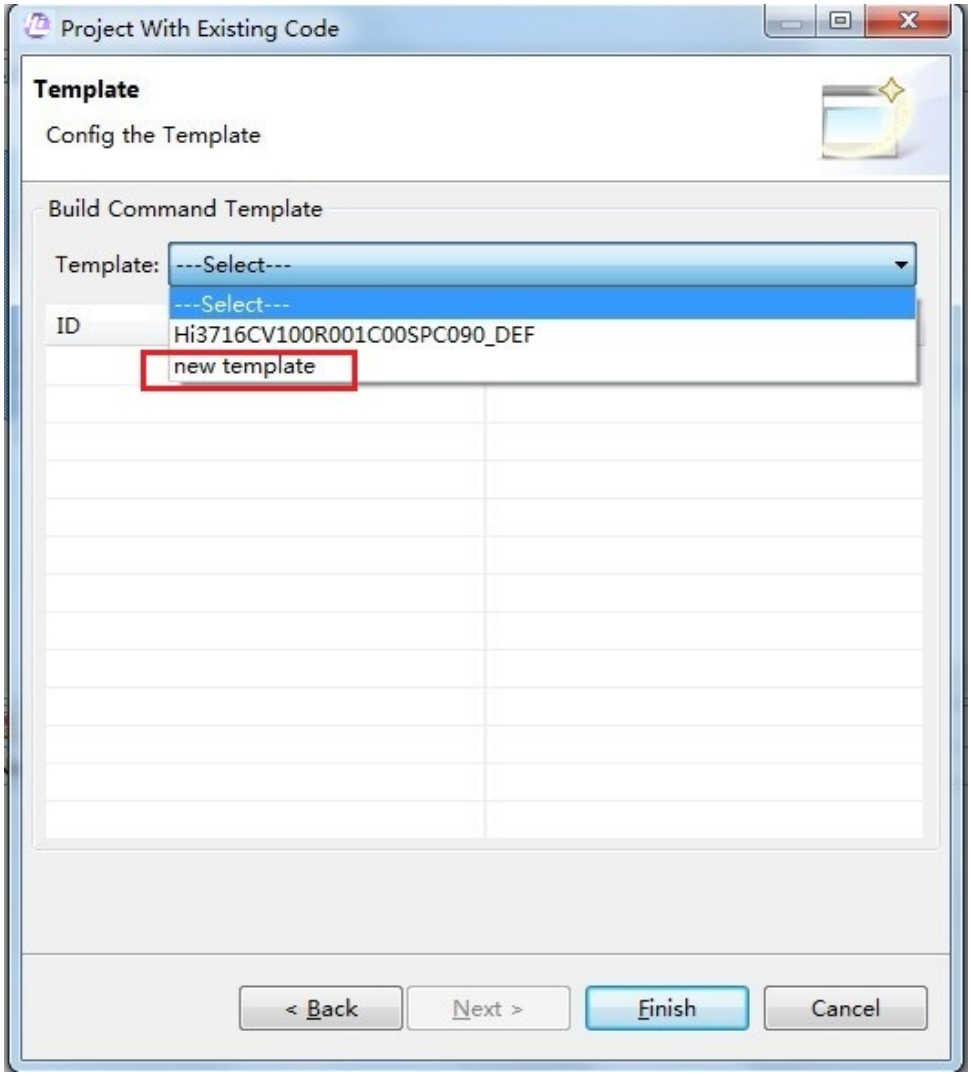
Figure 5-8 Adding a template



If you select an existing template from the **Template** drop-down list, add and remove commands by clicking **Add Row** and **Remove Row**, and click **Update** to apply the modification. You can also click **Delete** to delete the template.

When an application is being created, the newly added template can be selected, as shown in Figure 5-9.

Figure 5-9 Selecting a new template



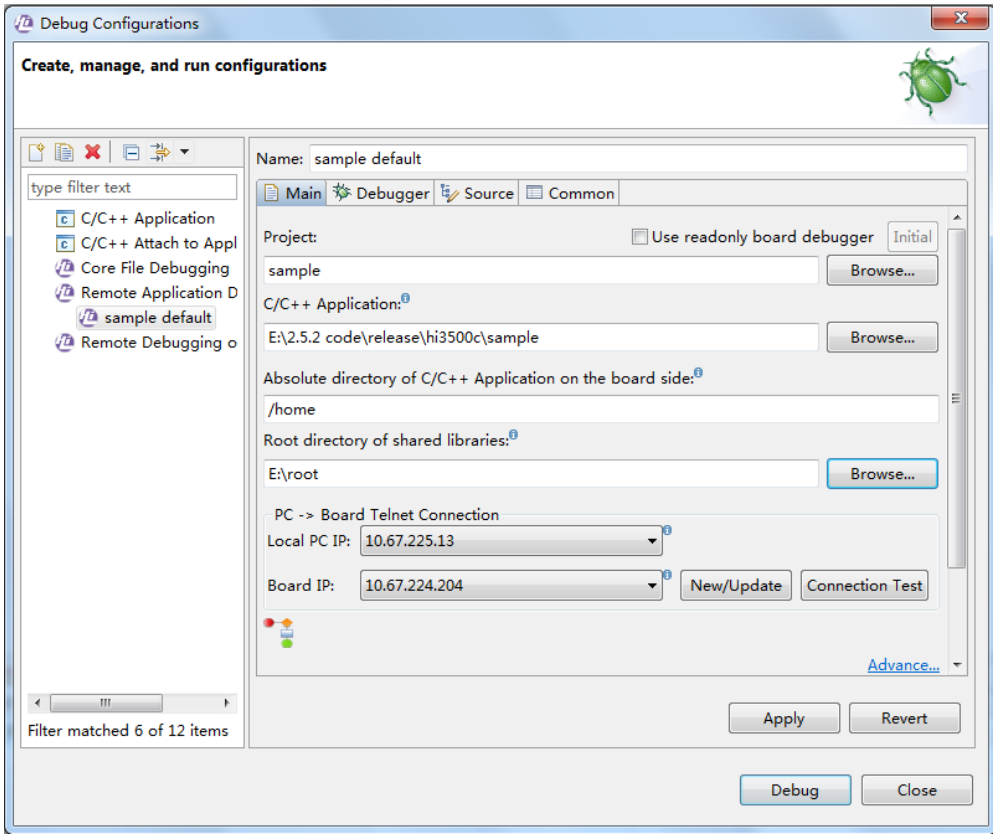
5.3 Debugging Configurations

5.3.1 Linux Application Debugging Configurations

5.3.1.1 Main Tab Page

Figure 5-10 shows the **Main** tab page for Linux application debugging configurations.

Figure 5-10 Main tab page for Linux application debugging configurations



- **Project**: project to which the application to be debugged belongs
- **C/C++ Application**: application to be debugged
- **Absolute directory of C/C++ Application on the board side**: directory on the target board to which the application is downloaded. The default value is **/home**.
- **Root directory of shared libraries**: directory on the PC for storing dynamic linked libraries. This item needs to be configured if you need to debug the source code in the dynamic linked library in a single step. This directory is equivalent to the root directory of the board for the dynamic linked library files in the directory.
- **Local PC IP**: IP address for the local PC
- **Board IP**: IP address for the target board

5.3.1.2 Advanced Settings

Before debugging, if you need to run commands on the target board, transfer parameters to the application to be debugged, or set environment variables on the target board, advanced settings on the **Main** tab page are required. Click **Advance** on the **Main** tab page, as shown in Figure 5-11. The dialog box shown in Figure 5-12 is displayed.

Figure 5-11 Clicking Advance on the Main tab page

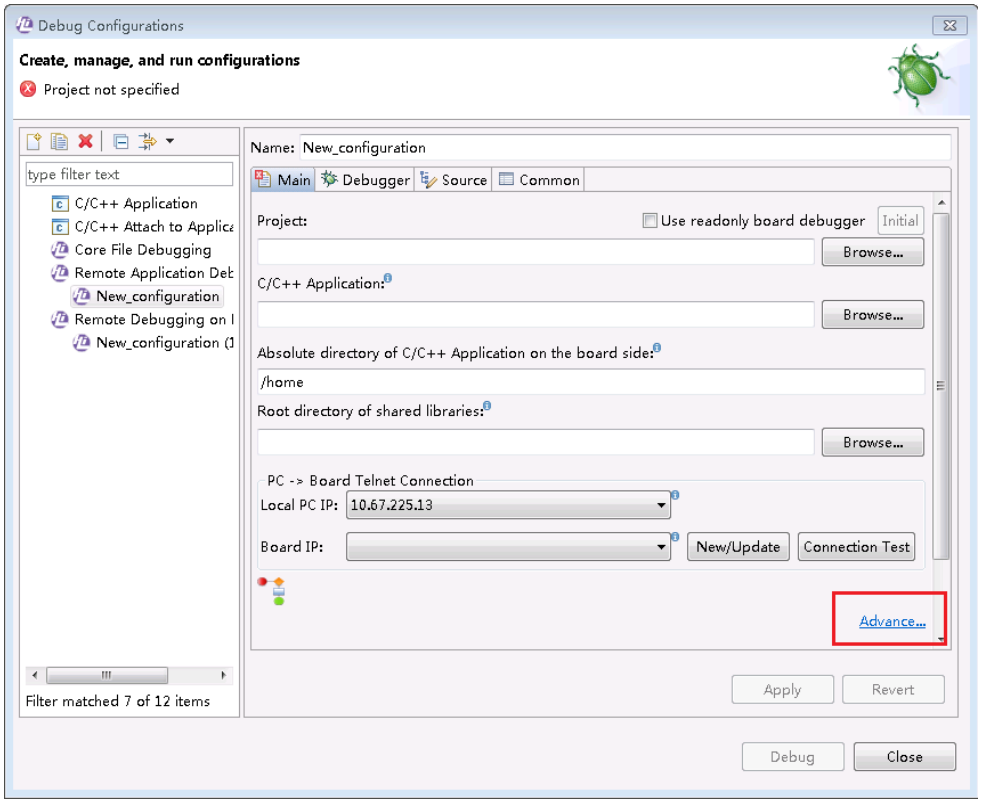
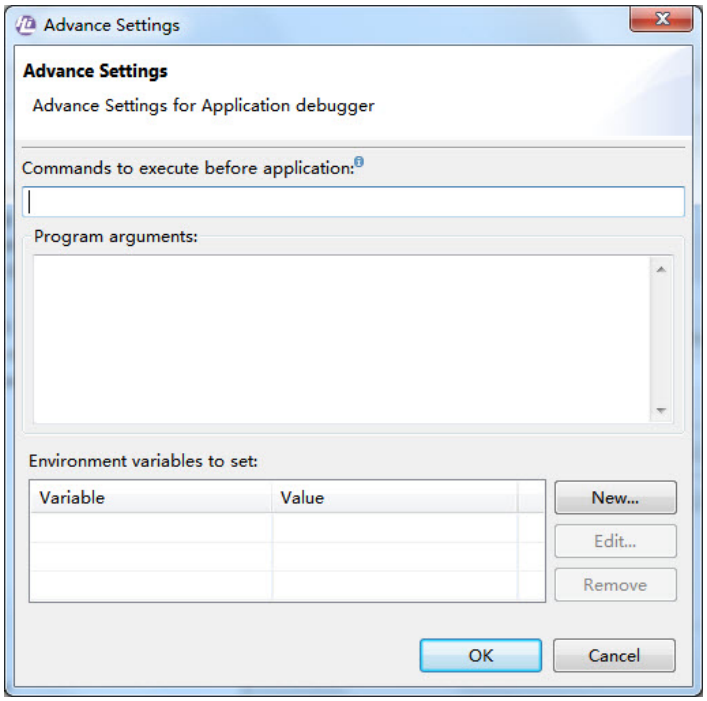


Figure 5-12 Advance Settings dialog box

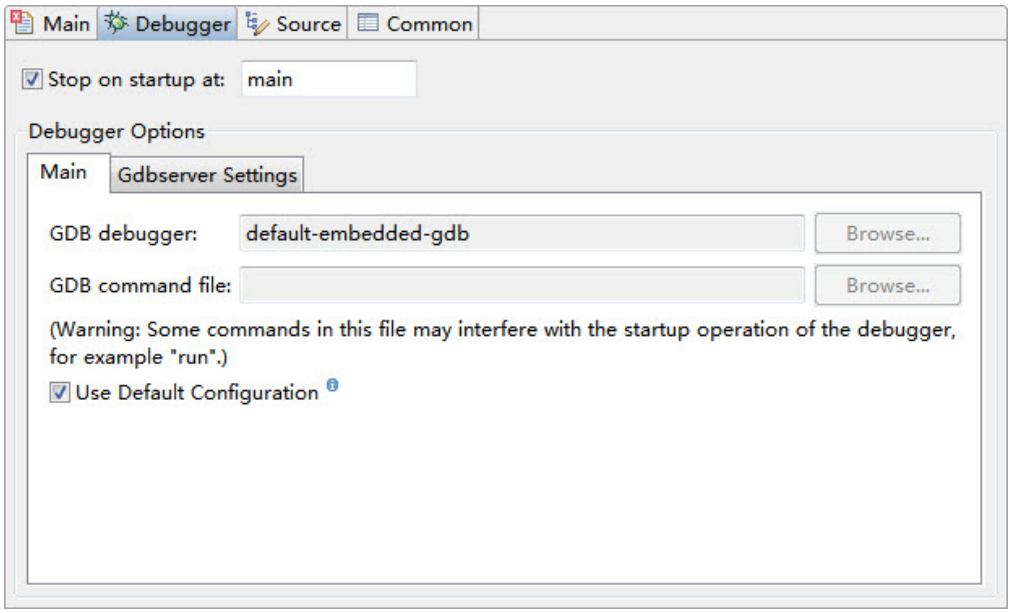


- **Commands to execute before application**: commands that need to be executed on the target board before debugging
- **Program arguments**: parameters to be transferred to the application
- **Environment variables to set**: environment variables to be configured on the target board

5.3.1.3 Debugger Tab Page

Figure 5-13 shows the **Debugger** tab page.

Figure 5-13 Debugger tab page

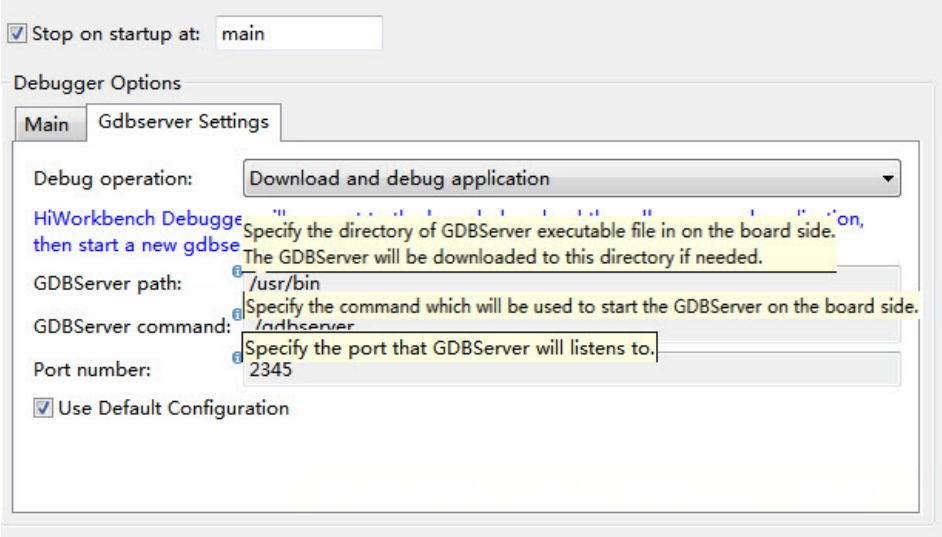


- **Stop on startup at**: The specified function name is considered as a breakpoint.
- **GDB debugger**: The specified GDB is used to debug the application at the background.
- **GDB command file**: The file contains the GDB commands, which are executed after the GDB starts.
- **Use Default Configuration**: If this option is not selected, you can specify a GDB. If this option is not selected, the embedded GDB is used by default.

5.3.1.4 Gdbserver Settings

Figure 5-14 shows **Gdbserver Settings**.

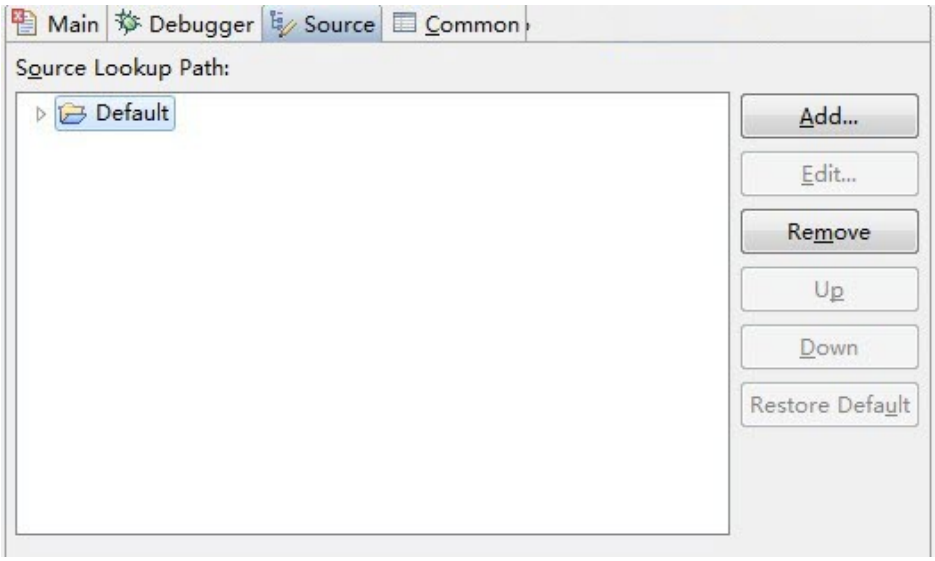
Figure 5-14 Gdbserver Settings



5.3.1.5 Source Tab Page

Figure 5-15 shows the **Source** tab page.

Figure 5-15 Source tab page

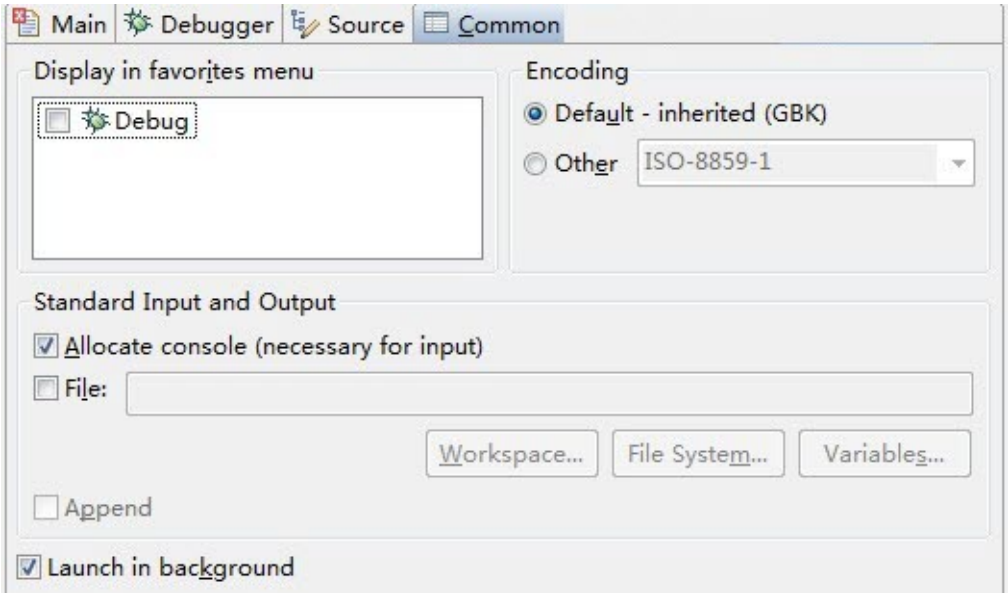


- **Add**: Adds new resources including the resource path search list.
- **Edit**: Modifies the selected resource.
- **Remove**: Deletes the selected item in the resource search list.
- **Up**: Moves the selected item upwards in the resource search list.
- **Down**: Moves the selected item downwards in the resource search list.
- **Restore Default**: Restores the default resource search list.

5.3.1.6 Common Tab Page

Figure 5-16 shows the **Common** tab page.

Figure 5-16 Common tab page



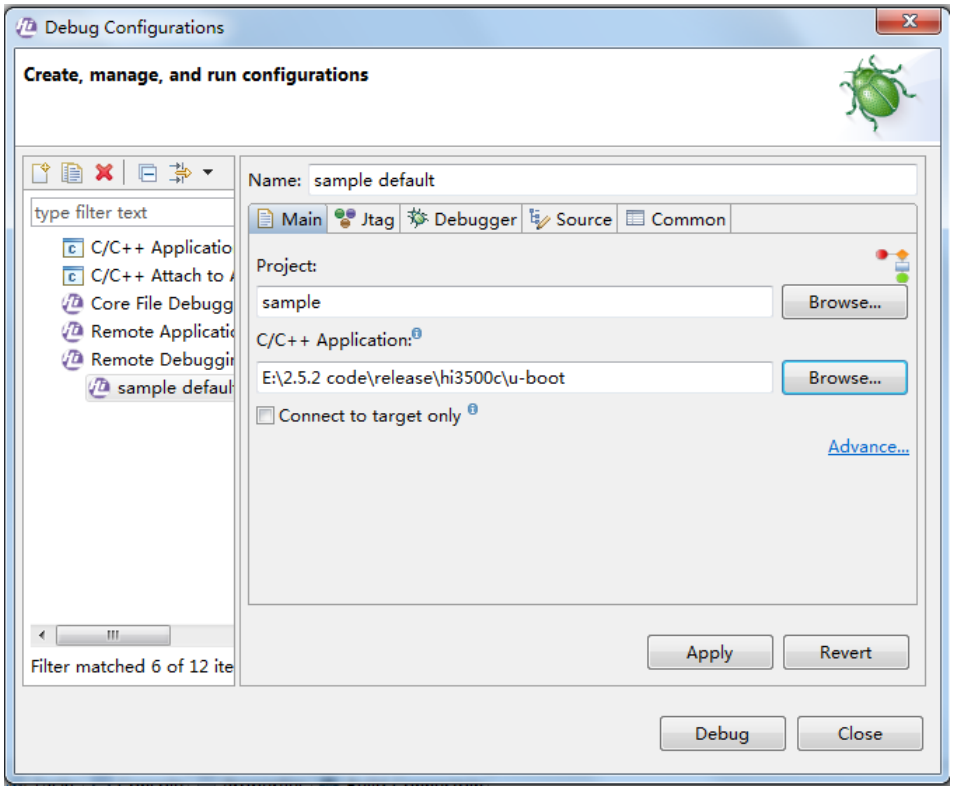
- **Display in favorites menu**: Specifies the names of configurations to be added to the operation or debugging menu to facilitate selection.
- **Encoding**: Specifies the encoding format used for console output.
- **Allocate console (necessary for input)**: If it is selected, a console view is allocated for receiving outputs.
- **File**: Specifies the name of the output file.
- **Workspace**: Specifies a file in the workspace to store the outputs.
- **File System**: Specifies a file in the file system to store the outputs.
- **Variables**: Specifies names of variables in the output file.
- **Append**: If it is selected, the file is re-created each time.
- **Launch in background**: If it is selected, the configuration is launched at the background.

5.3.2 Bare Board Program Debugging Configurations

5.3.2.1 Main Tab Page

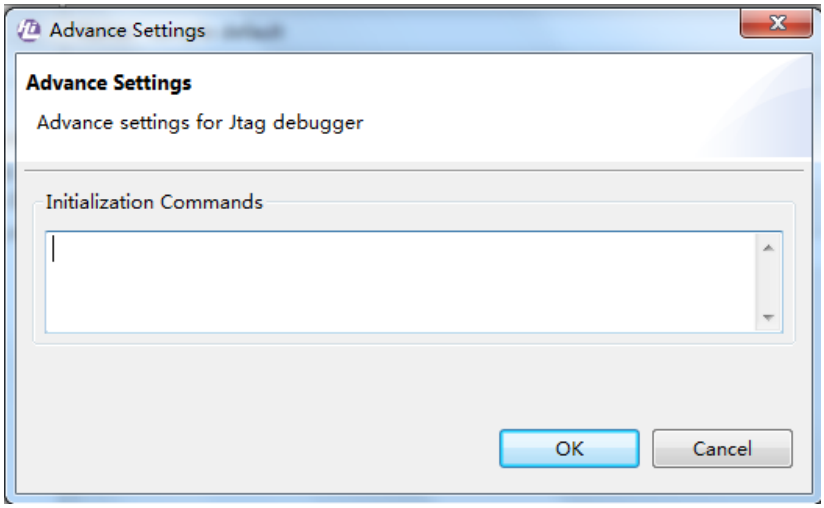
Figure 5-17 shows the **Main** tab page for bare board program debugging configurations.

Figure 5-17 Main tab page for bare board program debugging configurations



- **Project**: project to which the application to be debugged belongs
- **C/C++ Application**: C/C++ program to be debugged. Note that the selected executable program must be in ELF format.
- **Connect to target only**: This option can be selected if you want to connect to the target board directly by using the JTAG without loading the executable program. In this case, after the target board is connected, only the original code on the target board that is being executed can be view, only the assembly instructions can be viewed, and only instruction-level single-step debugging can be performed. If this option is not selected, the HiWorkbench automatically loads the executable file specified in **C/C++ Application** during JTAG connection. If the project contains source code, the source-code-level single-step debugging can be performed.
- **Advance**: Commands can be sent to the integrated OpenOCD by configuring the advanced settings before debugging, as shown in Figure 5-18. For details about the commands, see the *OpenOCD User's Guide*.

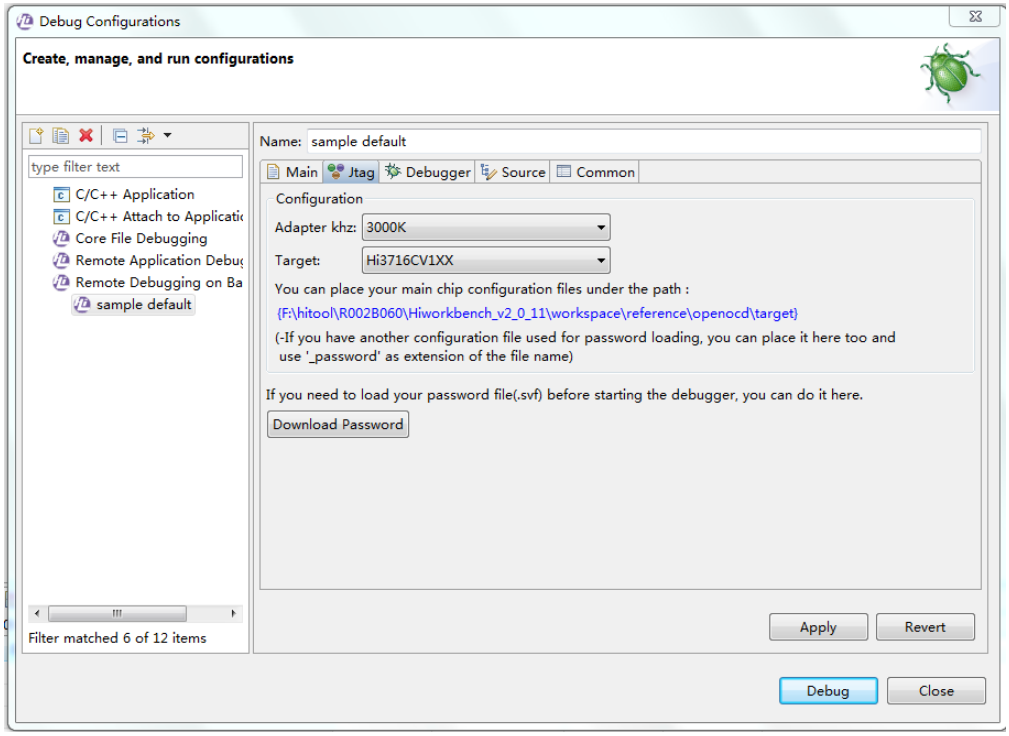
Figure 5-18 Advance Settings



5.3.2.2 Jtag Tab Page

Figure 5-19 shows the **Jtag** tab page.

Figure 5-19 Jtag tab page



- **Adapter khz**: JTAG connection frequency (500 kHz by default)
- **Target**: chip model of the target board

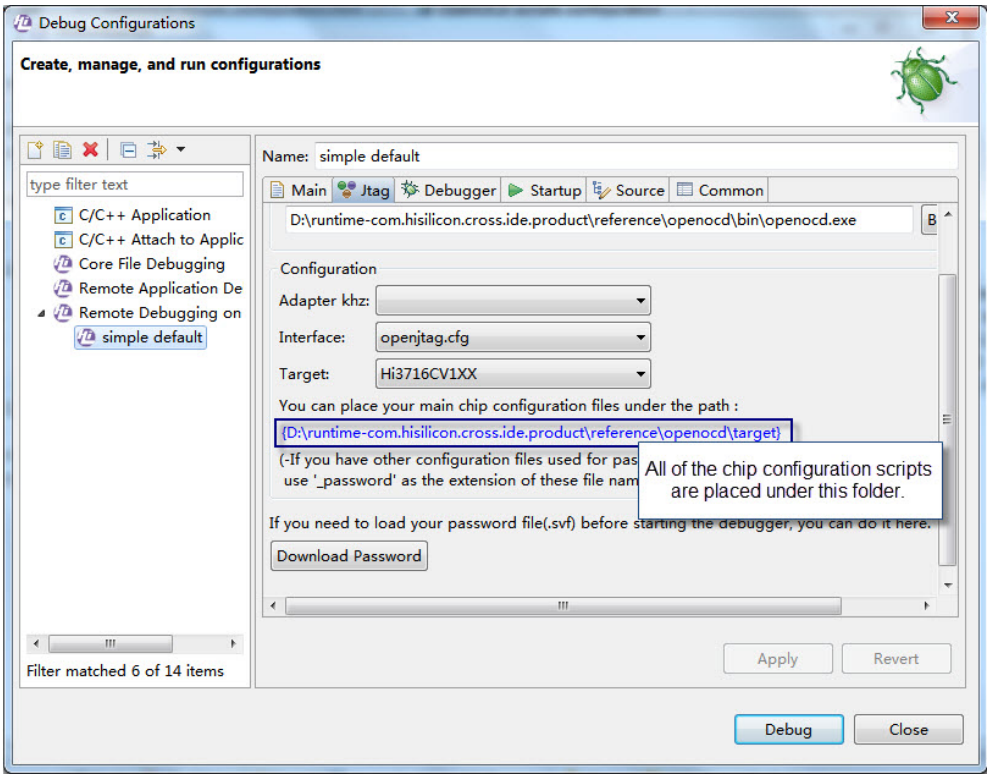
- **Download Password**: If the chip JTAG is locked using a keyword, you can click this button to download the SVF key file to unlock it.

5.3.2.3 Adding a Customized OpenOCD Script

To add a customized OpenOCD script, perform the following steps:

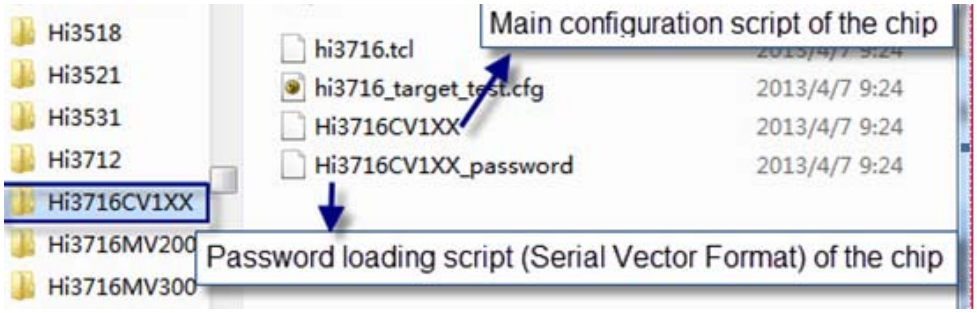
Step 1 Right-click the project name, and choose **Debug As** > **Debug Configurations**, or choose **Debug Configurations** from the drop-down list of the debug button on the toolbar. The **Debug Configurations** dialog box is displayed. Create a bare board configuration file, and view the path for the inherent script of the tool in the file system on the **Jtag** tab page, as shown in Figure 5-20.

Figure 5-20 OpenOCD script configuration



Step 2 Add a directory to the path viewed in step 1. The directory name will be displayed in the **Target** drop-down list. Save customized OpenOCD scripts in the directory. Note that the name of at least one configuration script file must be the same as the directory name (excluding the suffix). If there is a key file for unlocking the JTAG, **_password** must be used as the file name extension.

Figure 5-21 Configuration script directory structure



The HiWorkbench has some inherent chip configuration scripts. For details about the OpenOCD script, see the *OpenOCD User's Guide*.

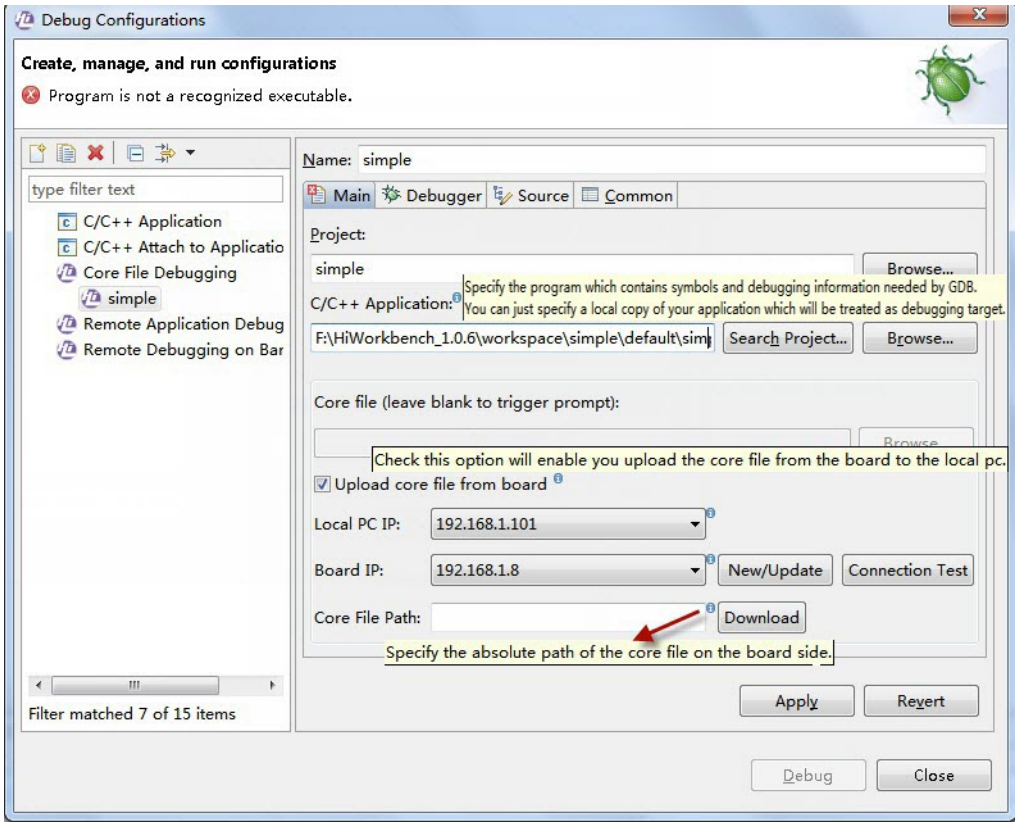
----End

5.3.3 Core File Debugging Configurations

5.3.3.1 Main Tab Page

Figure 5-22 shows the **Main** tab page for core file debugging configurations.

Figure 5-22 Main tab page for core file debugging configurations



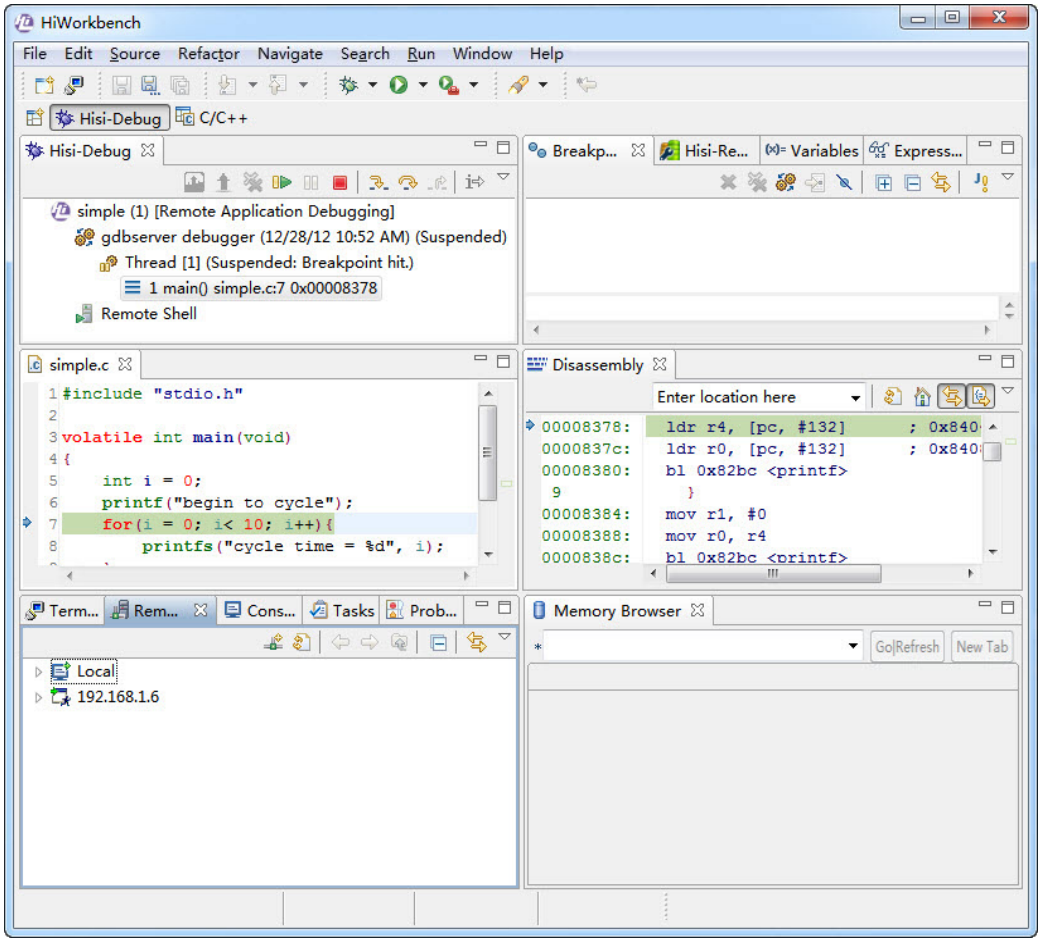
- **Project**: project to which the application to be debugged belongs
- **C/C++ Application**: application to be debugged
- **Core file (leave blank to trigger prompt)**: core file to be debugged on the PC
- **Upload core file from board**: If the core file to be debugged is on the target board but not the PC, you need to select this option to download the core file from the board for debugging.
- **Local PC IP**: IP address for the local PC. It is available when **Upload core file from board** is selected.
- **Board IP**: IP address for the target board. It is available when **Upload core file from board** is selected.
- **Core File Path**: absolute path of the core file on the target board. It is available when **Upload core file from board** is selected.

For details about the **Debugger**, **Source**, and **Common** tab pages, see section 5.3.1 "Linux Application Debugging Configurations."

5.4 Debugging Perspective View

After configuring parameters in the **Debug Configurations** dialog box, click **Debug** at the bottom of the dialog box. The debugging perspective view is displayed, as shown in Figure 5-23.

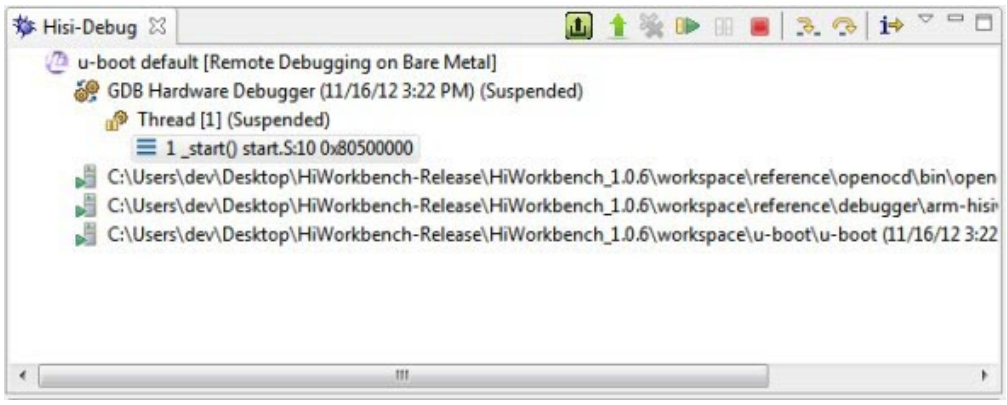
Figure 5-23 Debugging perspective view



5.4.2 Hisi-Debug

The **Hisi-Debug** tab page displays the tree structure of the target debugging information, as shown in Figure 5-24.

Figure 5-24 Hisi-Debug view



- : Released instance (configuration name and type)
- : Debugging instance (debugging name and status)
- : Thread instance (thread quantity and status)
- : Stack framework and instance (stack framework quantity, function, file name, and number of rows in the file)
- : Removes all stopped releases in the debugging view.
- : Restores the program.
- : Pauses the currently selected debugging target thread.
- : Ends the selected debugging session or progress. The impact depends on the type of the selected item in the debugging view.
- : Executes the current statement, but traces to the internal of the subprogram (step-by-step execution)
- : Executes the current statement, but skips the invocation of the subprogram (single-step execution)
- : (instruction step mode) If this button is clicked, the single-step and step-by-step operations are executed one by one according to the assembly instructions. Otherwise, they are executed according to the source code language mode.

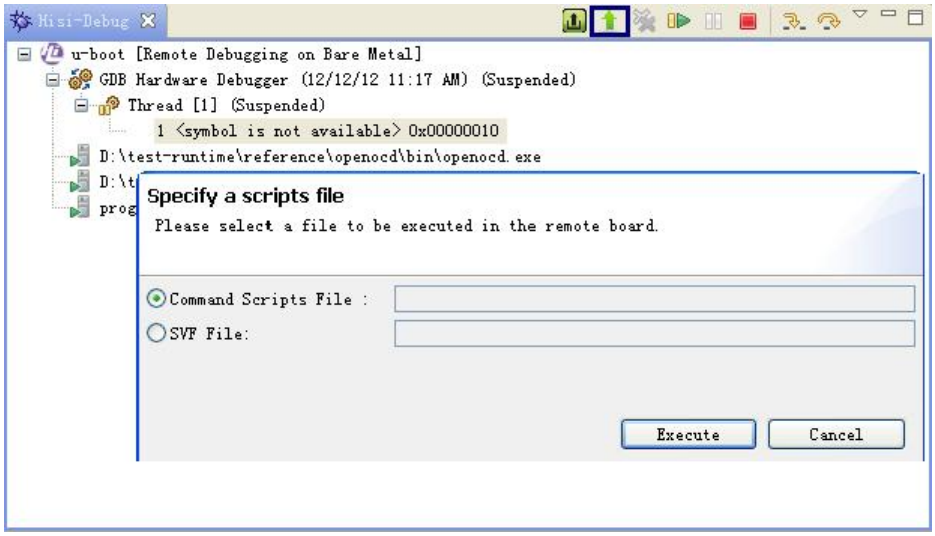
5.4.2.2 Executing a Script

During debugging of a bare board, you can run commands supported by the OpenOCD in batches or run the SVF script. This function can be used to initialize the DDR on the target board.

Step 1 Start bare board debugging.

Step 2 Click on the **Hisi-Debug** tab page. A dialog box is displayed, as shown in Figure 5-25.

Figure 5-25 Executing a script



Step 3 Specify the type of the script file to be executed, select a file, and click **Execute**.

----End

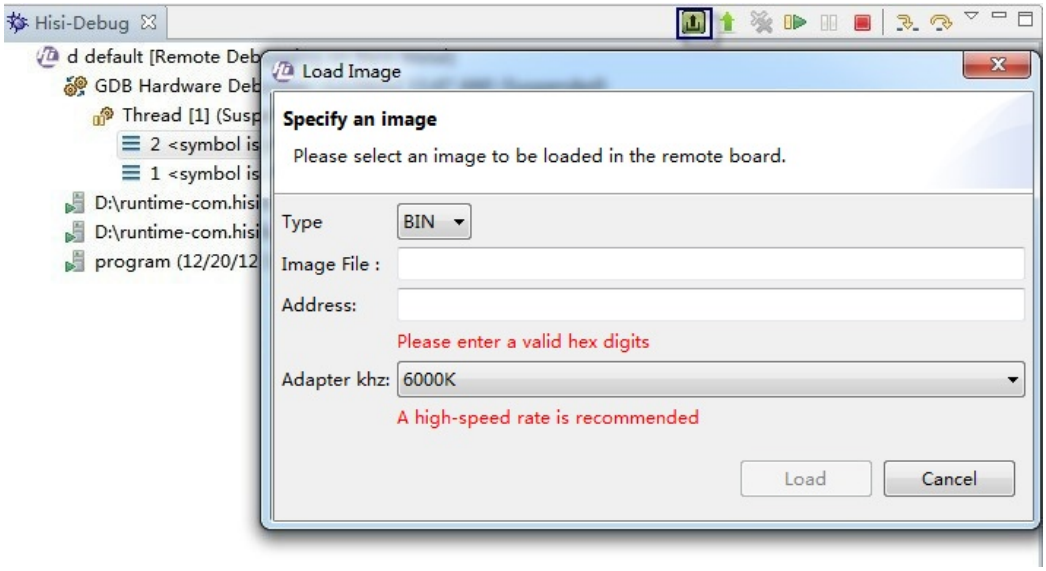
5.4.2.3 Loading an Image

During debugging of a bare board, you can download binary files to the memory space of the target board. To load an image, perform the following steps:

Step 1 Start bare board debugging.

Step 2 Click  . A dialog box is displayed, as shown in Figure 5-26.

Figure 5-26 Loading an image



Step 3 Specify the following information, and then click **Load** to load the image.

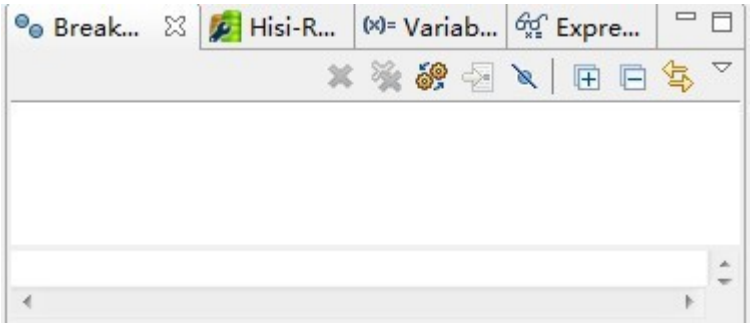
- **Type**: image file type (**BIN** or **ELF**)
- **Image File**: image to be downloaded to the target board
- **Address**: address on the target board for storing the downloaded image
- **Adapter khz**: download rate

----End

5.4.3 Breakpoints

The **Breakpoints** tab page lists all breakpoints in the current workspace. You can double-click a breakpoint to show its position in the editor (if the matched source code exists in the editor). You can also set available or unavailable breakpoints, or delete and add a breakpoint. See Figure 5-27.

Figure 5-27 Breakpoints tab page



The following describes tools on the toolbar. You can move the pointer over the icon to view the tool name.

- : Removes the selected breakpoint from the tab page.
- : Removes all breakpoints from the tab page.
- : Displays supported breakpoints only or not.
- : Searches for a file in the project based on the file name (opens the position of related breakpoints in the CDT editor).
- : Skips all breakpoints.
- : Expands all items on the tab page.
- : Collapses all items on the tab page.
- : Updates information on the **Breakpoints** tab page with that on the debugging view.

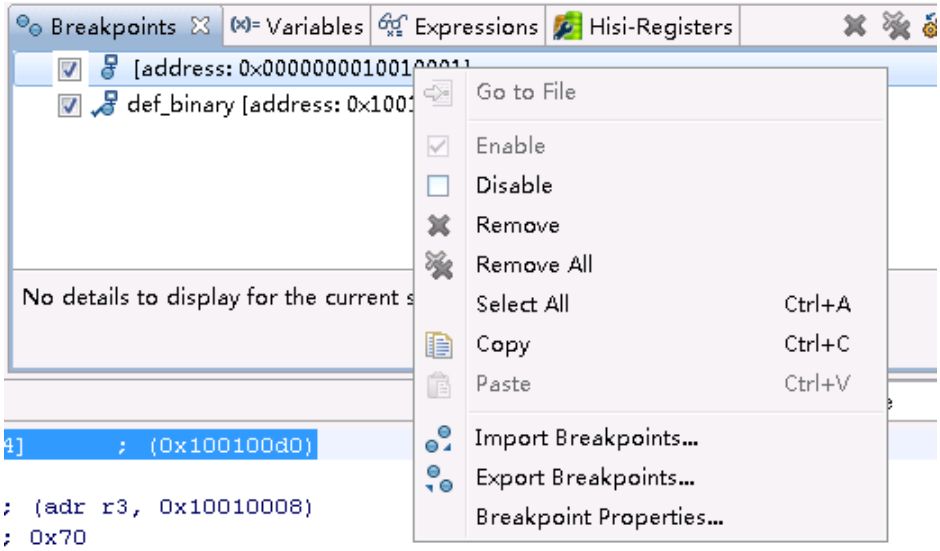
5.4.3.2 Setting Breakpoint Conditions

To set the breakpoint conditions, perform the following steps:

Step 1 Double-click a row to add a breakpoint. The added breakpoint can be viewed on the **Breakpoints** tab page.

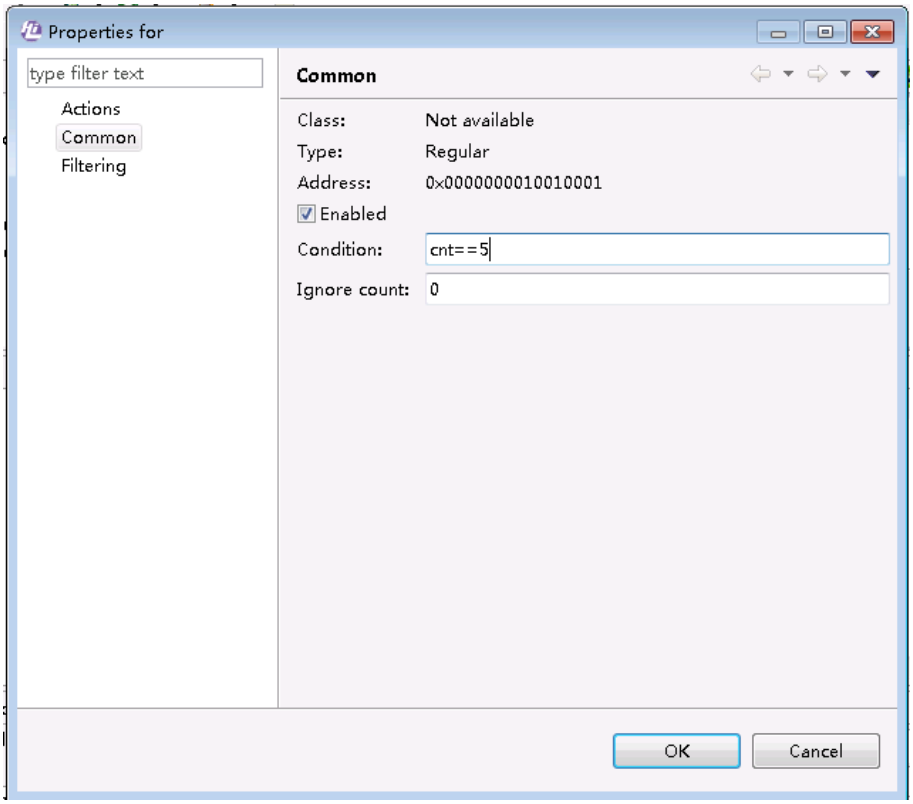
Step 2 Right-click the breakpoint on the **Breakpoints** tab page and choose **Breakpoint Properties**, as shown in Figure 5-28.

Figure 5-28 Shortcut menu on the Breakpoints tab page



Step 3 Choose **Common**, select **Enabled**, and enter the conditions (do not add "if"), as shown in Figure 5-29.

Figure 5-29 Setting the breakpoint conditions

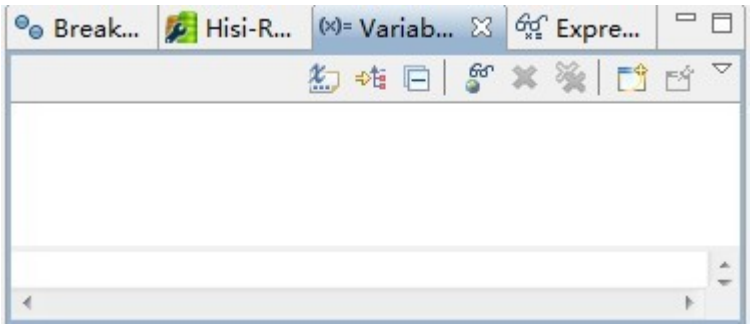


----End

5.4.4 Variables

The **Variables** tab page displays variables related to the stack structure in the debugging view. During the debugging of a program, more information about the selected variable is displayed in the detailed information panel. In addition, you can expand an object to display the details of each field. The variables are displayed in a column. The detailed information panel is displayed at the bottom of the tab page as a text box. See Figure 5-30.

Figure 5-30 Variables tab page

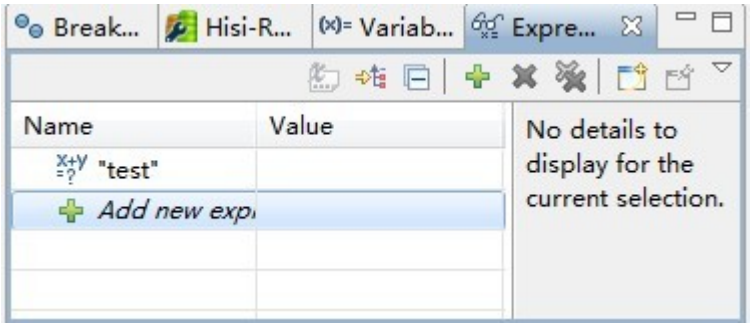


- : Displays the type name. The display type (for example, int) is related to each register value.
- : Displays the logical structure.
- : Collapses all current registration information.
- : Adds a global variable that applies to all workspaces.
- : Removes the selected variable.
- : Remove all variables.
- : Opens a new variable view.

5.4.5 Expressions

On the **Expressions** tab page, you can add or remove an expression or remove all expressions. See Figure 5-31.

Figure 5-31 Expressions tab page

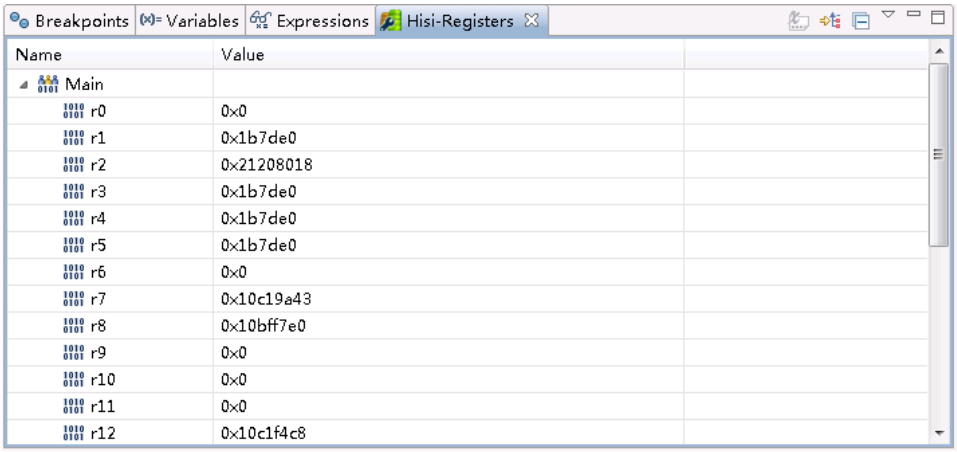


- : Displays the type name. The display type (for example, int) is related to each register value.
- : Displays the logical structure.
- : Collapses all current registration information.
- : Create an expression.
- : Removes the selected expression.
- : Removes all expressions.
- : Opens a new expression view.

5.4.6 Hisi-Registers

The **Hisi-Registers** tab page lists the current values of all registers in the chip ARM CPU, as shown in Figure 5-32. You can double-click the edit box next to the register to change the corresponding register value.

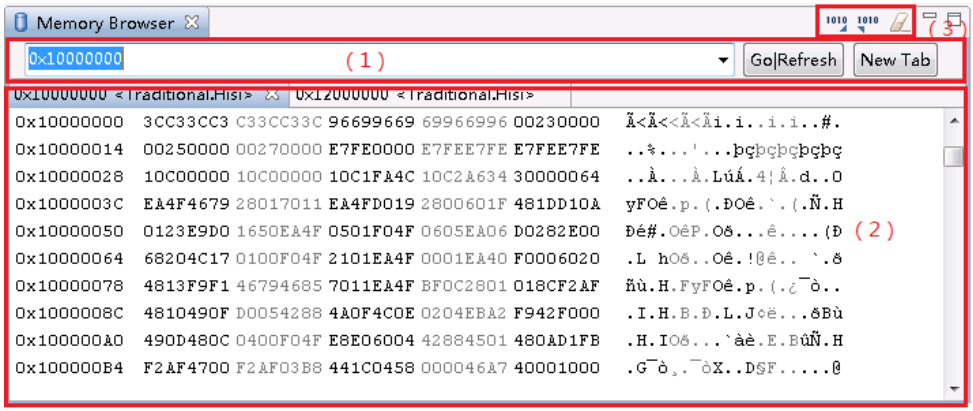
Figure 5-32 Hisi-Registers tab page



5.4.7 Memory Browser

The **Memory Browser** tab page allows you to monitor and modify the memory, as shown in Figure 5-33.

Figure 5-33 Memory browser



The **Memory Browser** tab page consists of three parts: address bar (area 1), memory content panel (area 2), and toolbar (area 3). Enter a base address in the address box (the address is automatically aligned). The corresponding memory content from the base address is displayed on the memory content panel. The memory content panel has multiple tabs, each of which corresponds to different addresses.

There are two buttons on the address bar:

- **Go|Refresh**: Refreshes memory contents.
- **New Tab**: Opens a new tab page.

Figure 5-34 shows the shortcut menu of the **Memory Browser** tab page.

Figure 5-34 Shortcut menu on the Memory Browser tab page

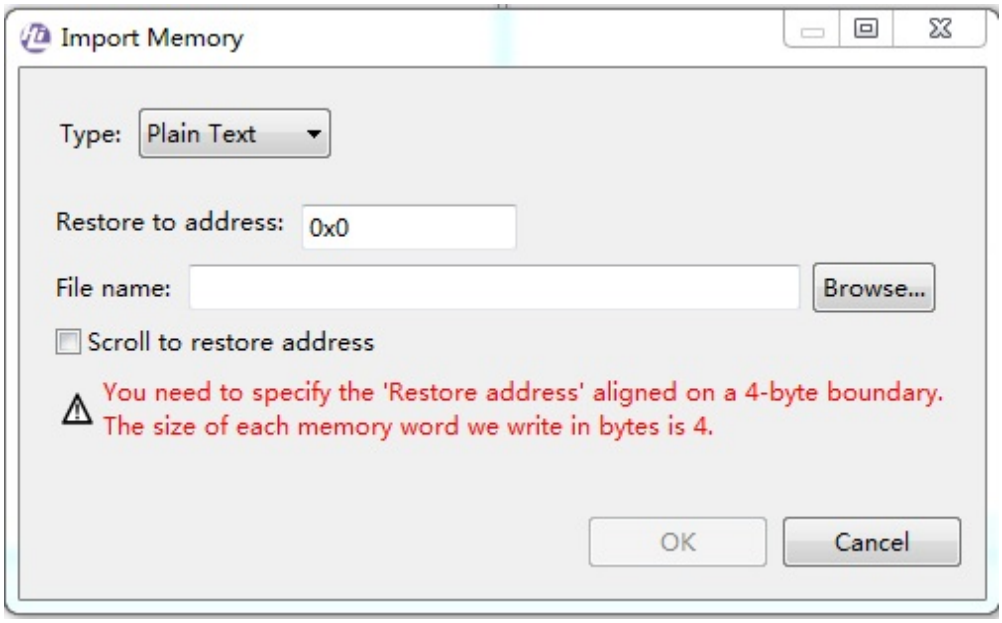
- **Panes**: The memory content panel consists of three parts. The leftmost column is the address column, the middle ones are the content columns (the number of content columns can be specified by configuring **Columns**), and the rightmost column displays texts corresponding to the memory values.

- **Endian**: Specifies the memory display mode (big-endian or little-endian).
- **Text**: Specifies character encoding for converting memory values into texts for display. It can be set to **ISO-8859-1** (default), **US-ASCII**, or **UTF-8**.
- **Cell Size**: Specifies the number of bytes displayed in the memory cell. It can be set to **1**, **2**, or **4** (default).
- **Radix**: Only the hexadecimal format is supported.
- **Columns**: Specifies the number of columns displayed in a row.
- **Copy**: Copies contents on the selected pane to the clipboard.

5.4.7.2 Importing Memory Data

Click . The **Import Memory** dialog box is displayed. Set the parameters and click **OK** to import the memory. See Figure 5-35.

Figure 5-35 Importing the memory



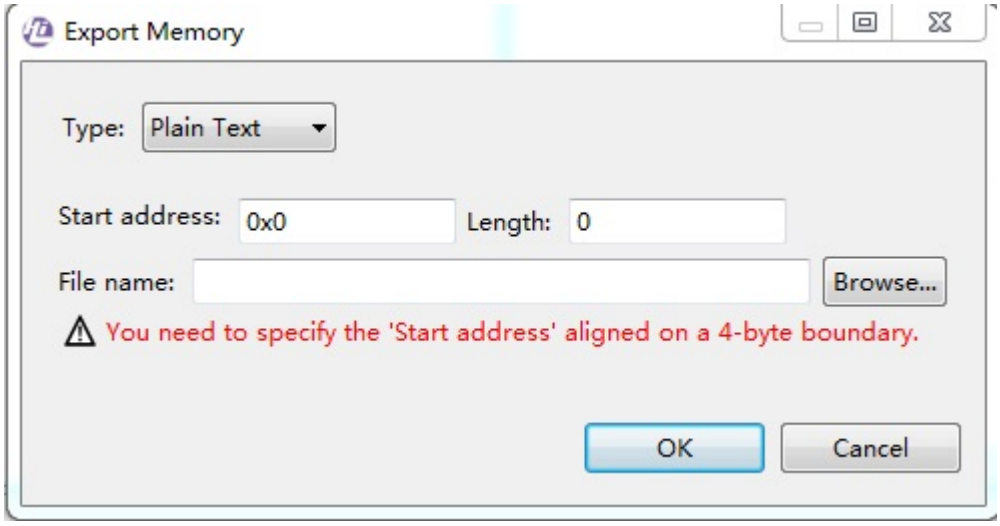
The parameters are described as follows:

- **Type**: text or binary
- **Restore to address**: a hexadecimal number starting with 0x
- **File name**: address for the file to be imported. The file must be selected based on the type. If the selected type is text, the selected file must be a text file. Otherwise, it must be a binary file.
- **Scroll to restore address**: If the option is selected, the tab page is rolled to the address specified in **Restore to address** after the import is complete.

5.4.7.3 Exporting Memory Data

Click . The **Export Memory** dialog box is displayed. Set the parameters and click **OK** to export the memory. See Figure 5-36.

Figure 5-36 Exporting the memory



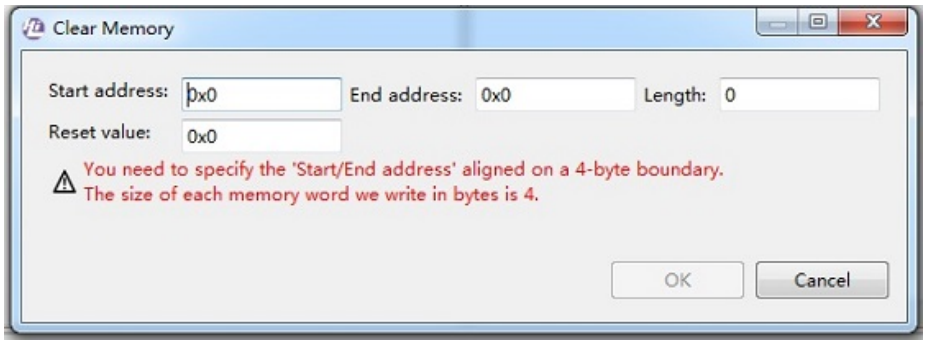
The parameters are described as follows:

- **Type**: text or binary
- **Start address**: a hexadecimal number starting with 0x
- **Length**: number of bytes of the memory to be exported
- **File name**: address for the file to be exported. The file must be selected based on the type. If the selected type is text, the selected file must be a text file. Otherwise, it must be a binary file.

5.4.7.4 Clearing Memory Data

Click  . The **Clear Memory** dialog box is displayed. Set the parameters and click **OK** to clear the memory. See Figure 5-37.

Figure 5-37 Clearing the memory



The parameters are described as follows:

- **Start address**: start address for the operation, a hexadecimal number starting with 0x
- **End address**: end address for the operation, a hexadecimal number starting with 0x

- **Length**: length of the memory to be cleared, automatically calculated based on the specified start address and end address
- **Reset value**: memory value for the clear operation, a hexadecimal number starting with 0x

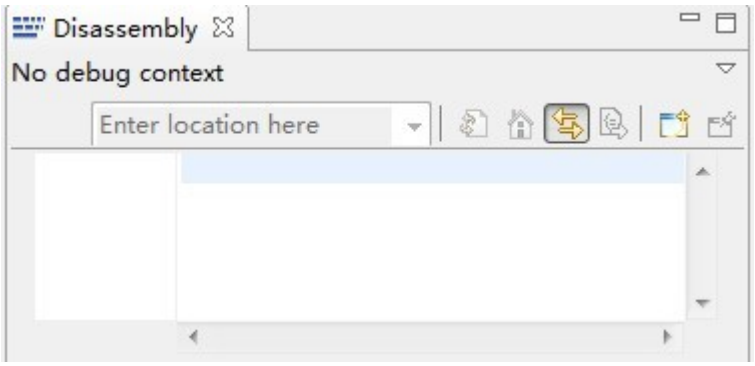
5.4.8 Disassembly

The **Disassembly** tab page displays the assembly instructions of the loaded program. The row that is being executed is marked with an arrow and highlighted. You can perform the following operations on this tab page:

- Set breakpoints in any positions.
- Enable and disable breakpoints and set their attributes.
- Control the program steps.
- Redirect to a specific instruction.

See Figure 5-38.

Figure 5-38 Disassembly tab page

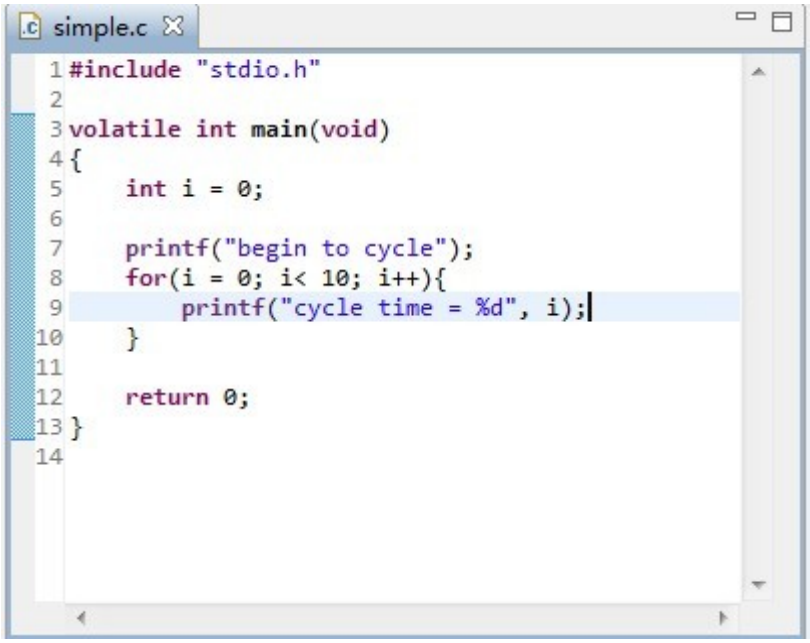


5.5 Code Editing

5.5.1 Code Editor

The code editor is located in the middle of the C/C++ perspective view. It is used to edit the C/C++ code. See Figure 5-39.

Figure 5-39 Code editor



The C/C ++ code editor supports the following functions:

- Syntax highlighted
- Content/Code assistance
- Code collapse
- Integrated debugging feature

5.5.2 Searching

Click  on the toolbar, or choose **Search** > **Search**, as shown in Figure 5-40, or press **Ctrl+H**.

Figure 5-40 Opening the search dialog box

The following search types are supported:

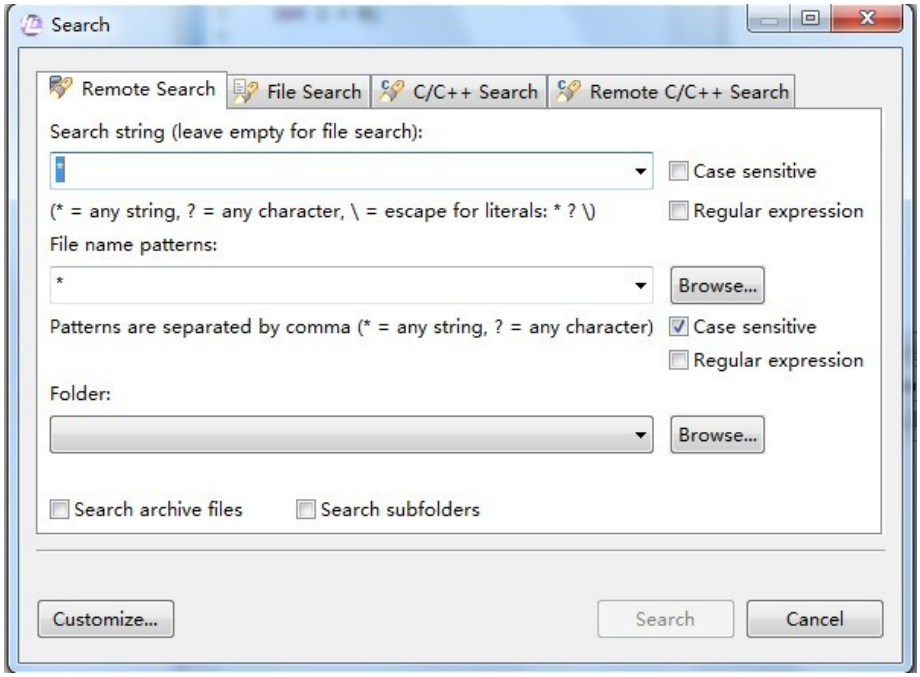
- Remote Search

- File Search
- C/C++ Search
- Remote C/C++ Search
- Text

5.5.2.2 Remote Search

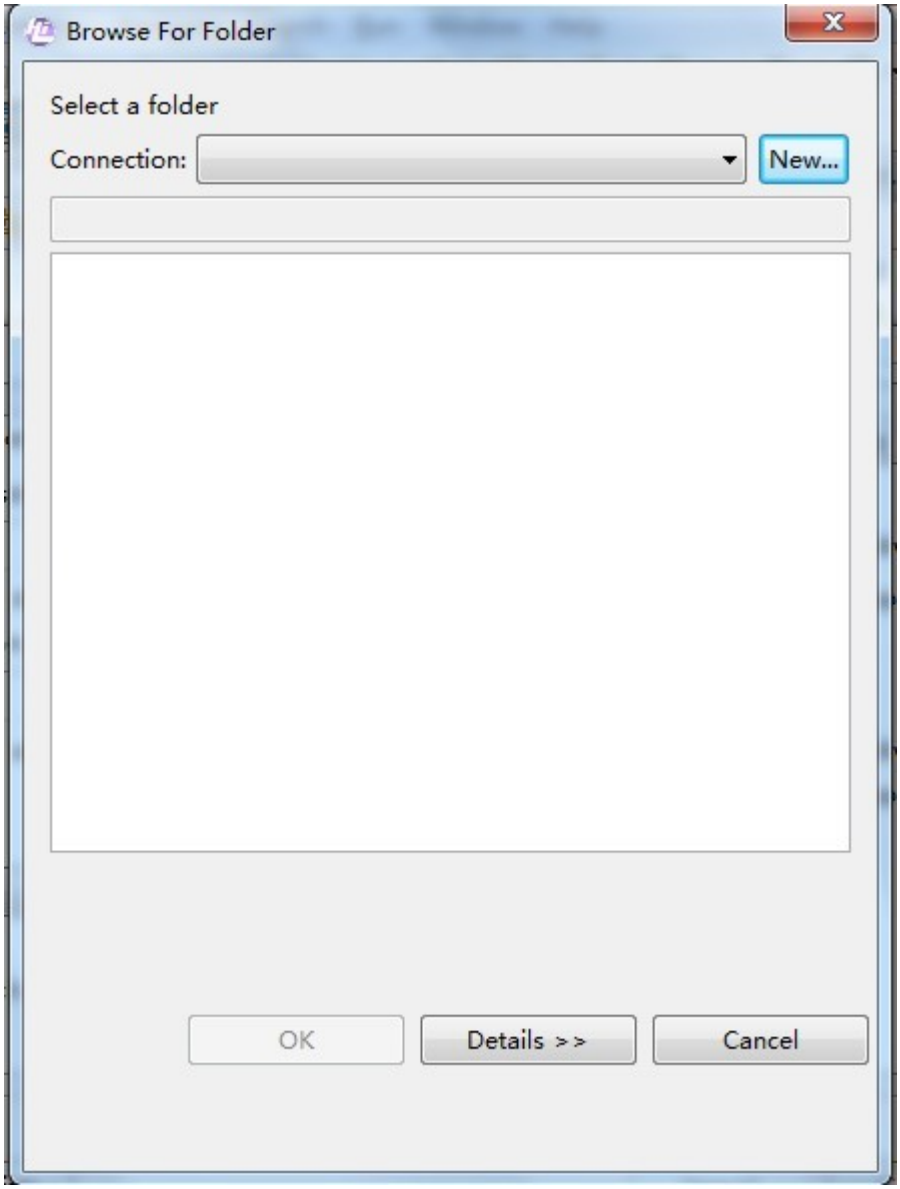
On the **Remote Search** tab page, you can search for remote files. See Figure 5-41.

Figure 5-41 Remote Search



- **Search string**: character string to be searched for. The following wildcards are supported:
  - * indicates any character string.
  - ? indicates any single character.
  - * is used to search for the asterisk (*).
- **File name patterns**: character string for filtering the file name.
- **Case sensitive**: whether the character string to be searched for is case sensitive.
- **Regular expression**: whether the character string to be searched for is a regular expression
- **Folder**: directory to be searched. Click **Browse**, and select a directory from the current connection in the displayed dialog box, or change the connection by selecting from the **Connection** drop-down list. Once the connection is changed, the directory list is updated accordingly. You can click **New** to create a connection. For details about how to create a connection, see the board connection management section.

Figure 5-42 Selecting a directory

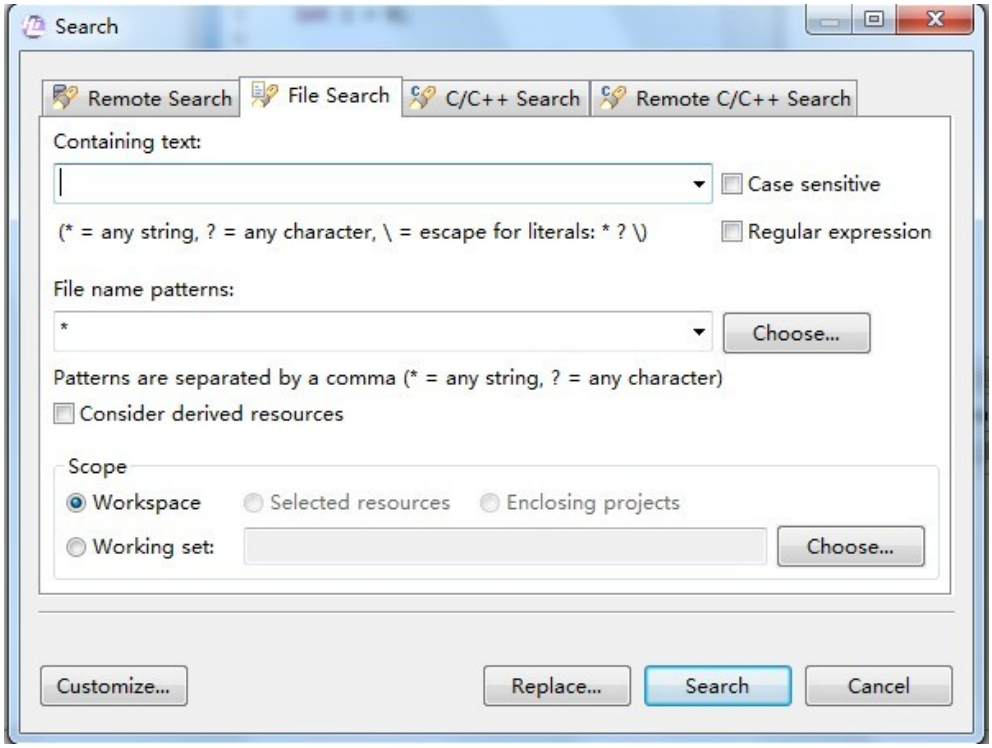


- **Search archive files**: whether to search compressed files (jar or tar files)
- **Search subfolders**: whether to search subdirectories

5.5.2.3 File Search

The **File Search** tab page is used to search for files. See Figure 5-43.

Figure 5-43 File Search

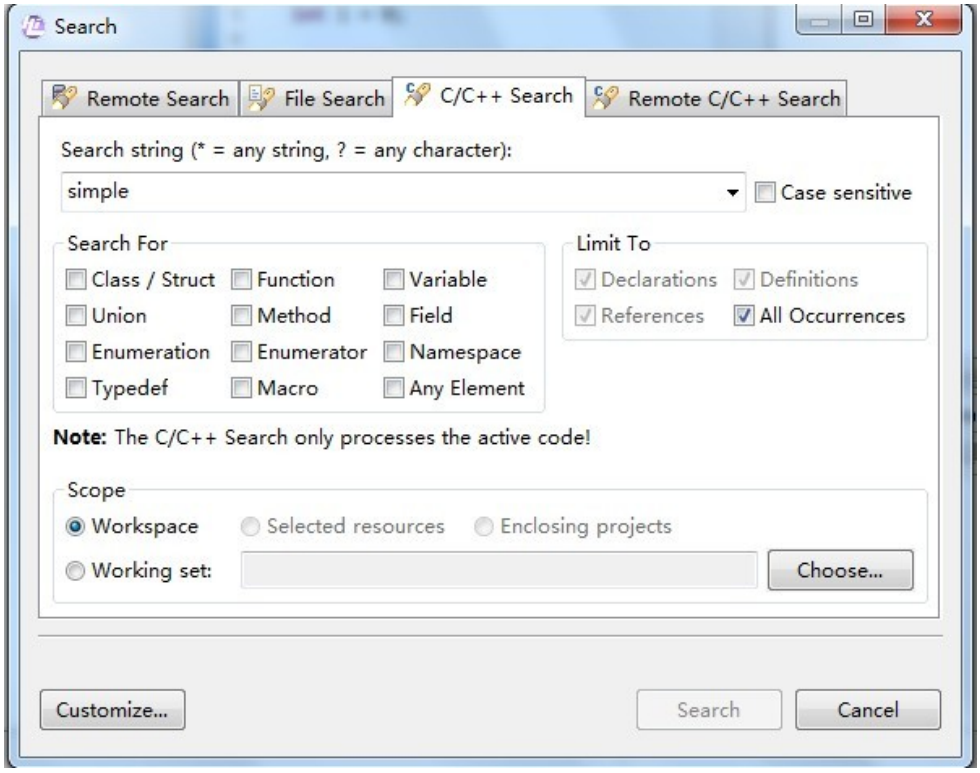


- **Containing text**: a character string to be searched for. The following wildcards are supported:
  - * indicates any character string.
  - ? indicates any single character.
  - ***** is used to search for the asterisk (*).
- **Case sensitive**: whether the character string to be searched for is case sensitive
- **Regular expression**: whether the character string to be searched for is a regular expression
- **File name patterns**: filtering file name
- **Scope**: search scope
  - **Workspace**: searches the entire workspace.
  - **Selected resources**: searches the selected resources in the project view.
  - **Enclosing projects**: searches selected resources in a closed project (including the path).
  - **Working set**: searches in the working set.

5.5.2.4 C/C++ Search

Figure 5-44 shows the **C/C++ Search** tab page.

Figure 5-44 C/C++ Search



- **Search string**: character string to be searched for. The following wildcards are supported:
  - * indicates any character string.
  - ? indicates any single character.
  - * is used to search for the asterisk (*).
- **Case sensitive**: whether the character string to be searched for is case sensitive
- **Search For**: type of elements to be searched for
  - **Class/Struct**: Includes classes and structures in the search.
  - **Function**: Searches for global functions or functions in a namespace (functions that are not members of a class, structure, or union).
  - **Variable**: Searches for variables that are not members of a class, structure, or union.
  - **Union**: Searches for unions.
  - **Method**: Searches for methods that are members of a class, structure, or union.
  - **Field**: Searches for fields that are members of a class, structure, or union.
  - **Enumeration**: Searches for enumerations.
  - **Enumerator**: Searches for enumerators.
  - **Namespace**: Searches for namespaces.
  - **Typedef**: Searches for type definitions.
  - **Macro**: Searches for macros.
  - **Any Element**: Searches for all elements.

- **Limit to**: type of context to be searched for
  - **Declarations**: Limits the search to declarations.
  - **Definitions**: Limits the search to definitions (for functions, methods, variables, and fields).
  - **References**: Limits the search to references.
  - **All Occurrences**: Includes declarations, definitions, and references in the search.
- **Scope**: search scope
  - **Workspace**: searches the entire workspace.
  - **Selected resources**: searches the selected resources in the project view.
  - **Enclosing projects**: searches selected resources in a closed project (including the path).
  - **Working set**: searches in the working set.

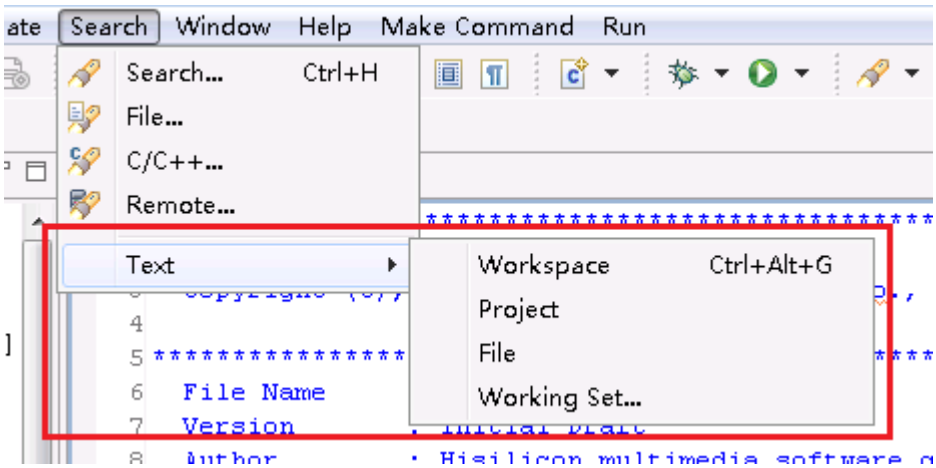
5.5.2.5 Remote C/C++ Search

Remote C/C++ search is the same as C/C++ search.

5.5.2.6 Text Search

Select the text to be searched for in the code editor, and choose **Search** > **Text**. Then the selected text can be searched for in the workspace, project, file, or working set. See Figure 5-45.

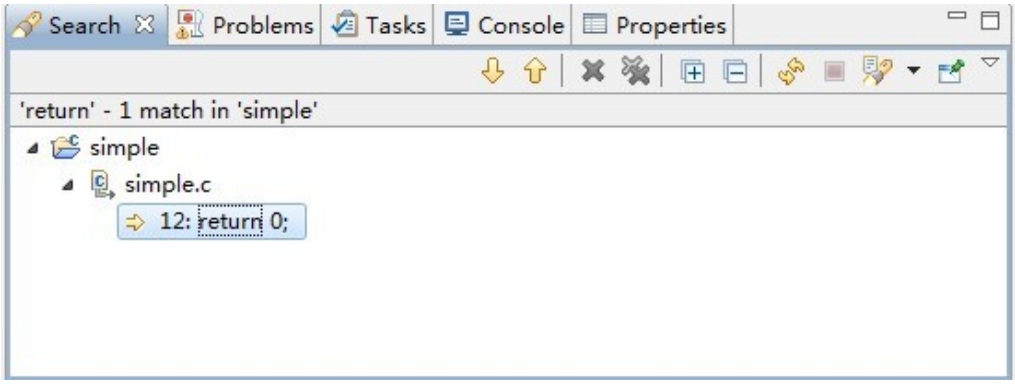
Figure 5-45 Searching for texts



5.5.2.7 Search Tab Page

The **Search** tab page displays the search result. Double-click the result. The corresponding file position is located, as shown in Figure 5-46.

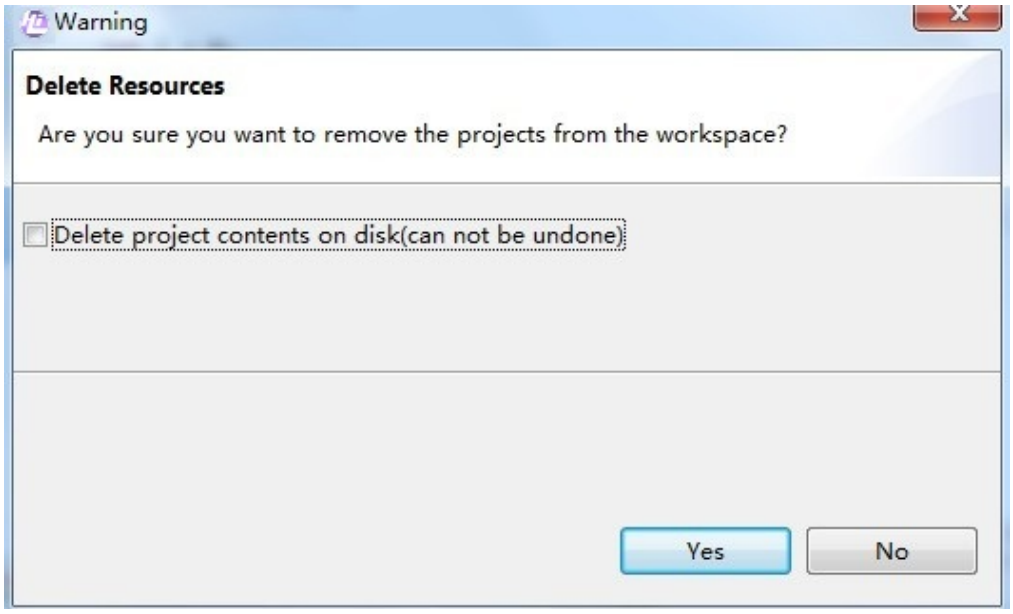
Figure 5-46 Search tab page



5.5.3 Deleting Resources

Right-click the project name in **Project Explorer**, and choose **Delete** from the shortcut menu. The confirmation dialog box is displayed, as shown in Figure 5-47. If you select **Delete project contents on disk (cannot be undone)**, all the files in the project are permanently deleted from the hard disk.

Figure 5-47 Deletion confirmation dialog box



5.5.4 Indexer

Choose **Windows** > **Preferences**. The **Preferences** dialog box is displayed. Expand **C/C ++**, and click **Indexer**, as shown in Figure 5-48. Select **Enable indexer**. Then you can set parameters of the indexer.

Figure 5-48 Indexer parameters

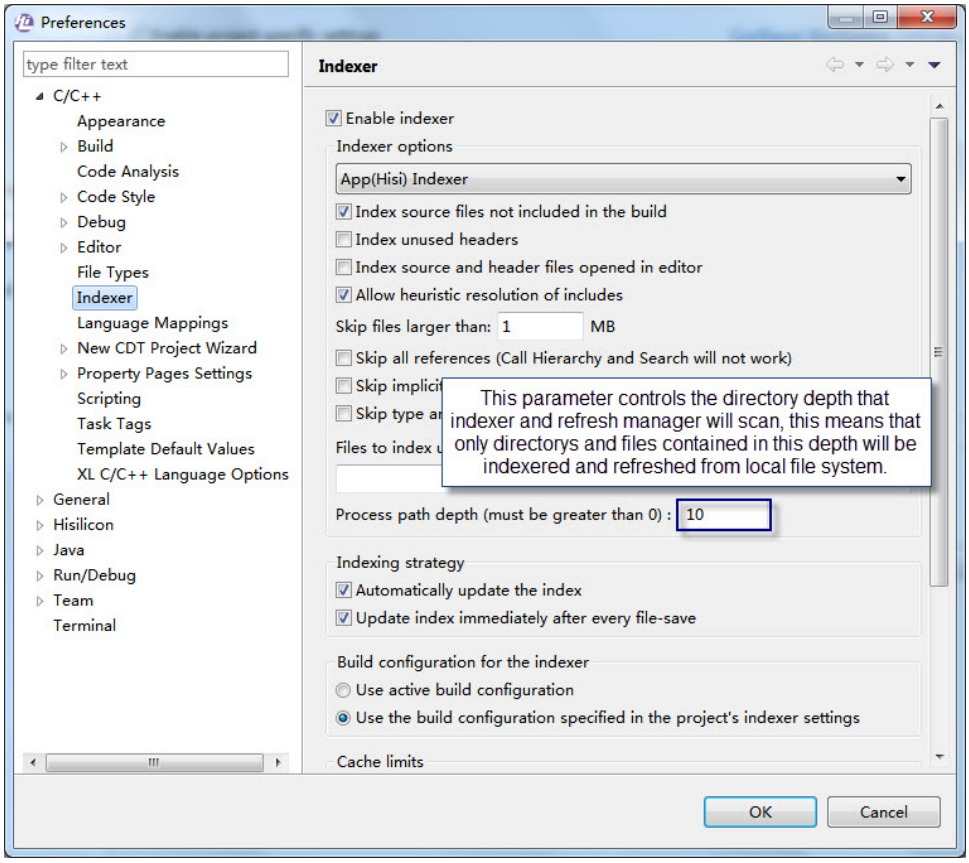
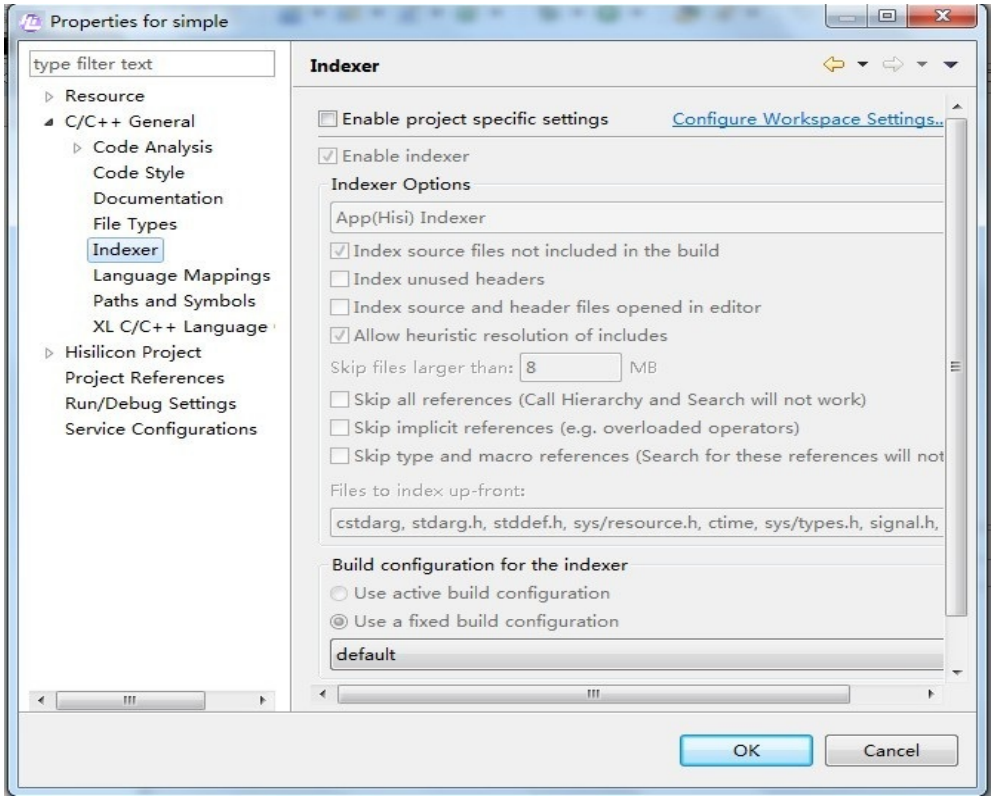


Table 5-1 Indexer options

Option	Description
Select Indexer	Select the indexer to use by default for all new projects.
Index source files not included in the build	Activate this checkbox to index all source files used by the project.
Index unused headers	Activate this checkbox to index unused header files.
Allow heuristic resolution of includes	Activate this checkbox to allow the indexer to skip indexing duplicate include files.
Skip files larger than	Enter the maximum file size to skip when indexing.
Skip all references (Call Hierarchy and Search will not work)	Activate this checkbox to not index references to save space and gain performance. This prevents some portions of CDT from working, like Call Hierarchy and search for references.
Skip implicit references (e.g. overloaded operators)	Activate this checkbox to not index implicit references.
Skip type and macro references (Search for these references will not work)	Activate this checkbox to not index macro or type references. This prevents some portions of CDT from working, like Search.
Files to index up-front	Type a comma separated list of files that should always be indexed immediately.
Automatically update the index	Activate this checkbox to have the index automatically update as it requires.
Update index immediately after every file change	Activate this checkbox to force an index update whenever a file is saved.
Use active build configuration	Activate to always use the active build configuration indexer settings to build the index.
Use the build configuration specified in the project's indexer settings	Activate to always use the project's indexer settings to build the index.
Index database cache: Limit relative to the maximum heap size	Specify the relative size limit the index can reach based on the maximum heap size.
Index database cache: Absolute limit	Specify the maximum size the index database cache is limited to.
Header file cache: Absolute limit	Specify the maximum size the header file cache is limited to.

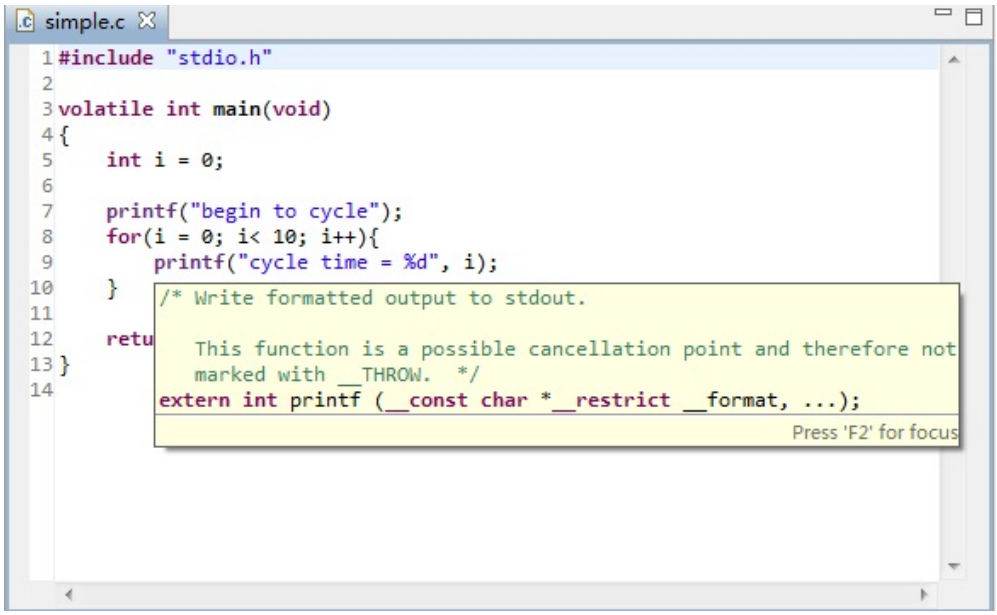
Right-click the project name and choose **Properties** from the shortcut menu. Expand **C/C ++ General**, and click **Indexer**. Then you can set the indexer for the project. See Figure 5-49.

Figure 5-49 Project indexer



Once the indexer is activated, you can view the detail information when you move the pointer onto a variable or function, and you can press **F3** to redirect to the source code of the function. See Figure 5-50.

Figure 5-50 Detailed information



5.6 Resource Filter

The resource filter is used to set the type of files that can and cannot be displayed in **Project Explorer**.

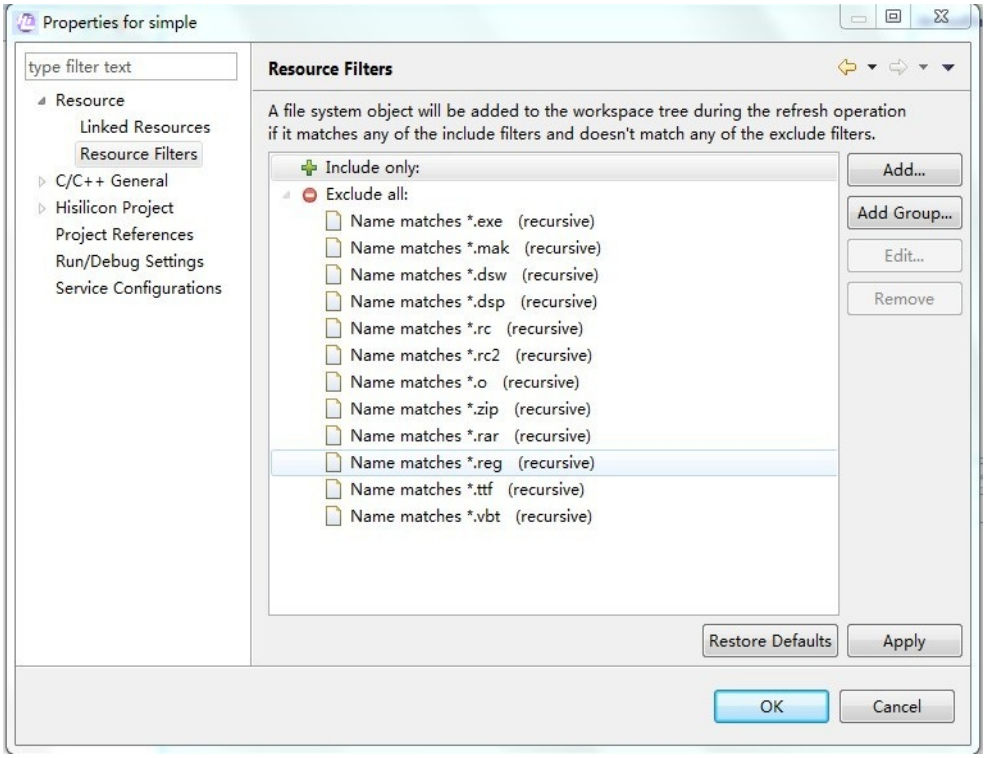
If the resource filter is added to a project or directory, unnecessary files or directories are not displayed in **Project Explorer**, which improves the performance of the HiWorkbench.

The file filter can filter files or directories.

The **Include only** resource filter allows only the files or directories that meet filtering conditions to be displayed in **Project Explorer**. The **Exclude only** resource filter allows only the files or directories that do not meet filtering conditions to be displayed in **Project Explorer**.

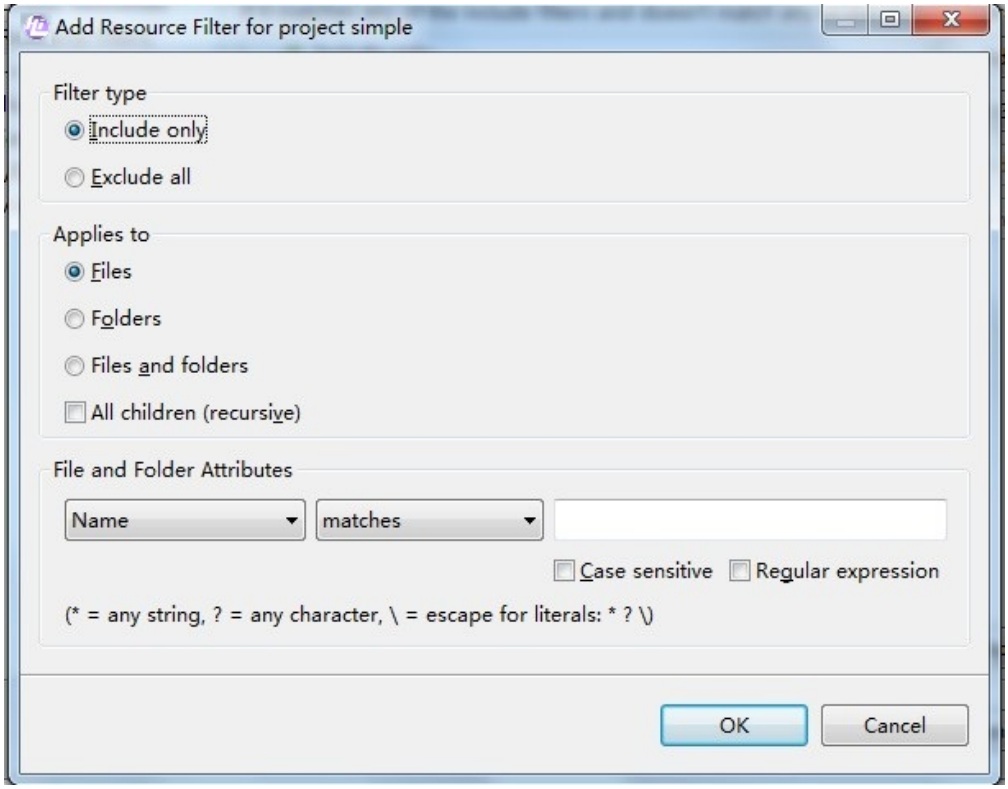
See Figure 5-51.

Figure 5-51 Resource Filters



Click **Add**. The **Add Resource Filter for project simple** dialog box is displayed, as shown in Figure 5-52. Set the filter type, target, and target attributes.

Figure 5-52 Adding a filter

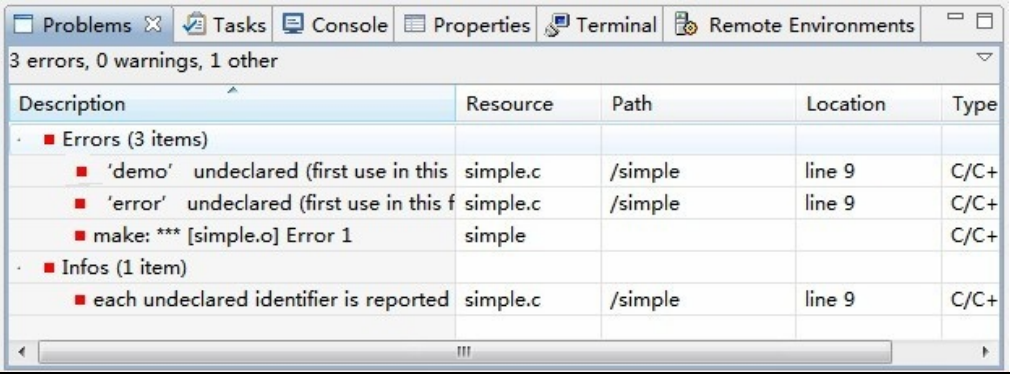


5.7 Error Parser

5.7.1 Problems Tab Page

The **Problems** tab page displays alarms (error, warning, and information levels) reported by the tool chain during compilation to facilitate fault location, as shown in Figure 5-53. The displayed information includes the alarm type, the source code file and the corresponding path, and the source code row and type information. Double-click an item in the list. Then the editor automatically locates the corresponding code row.

Figure 5-53 Problems tab page

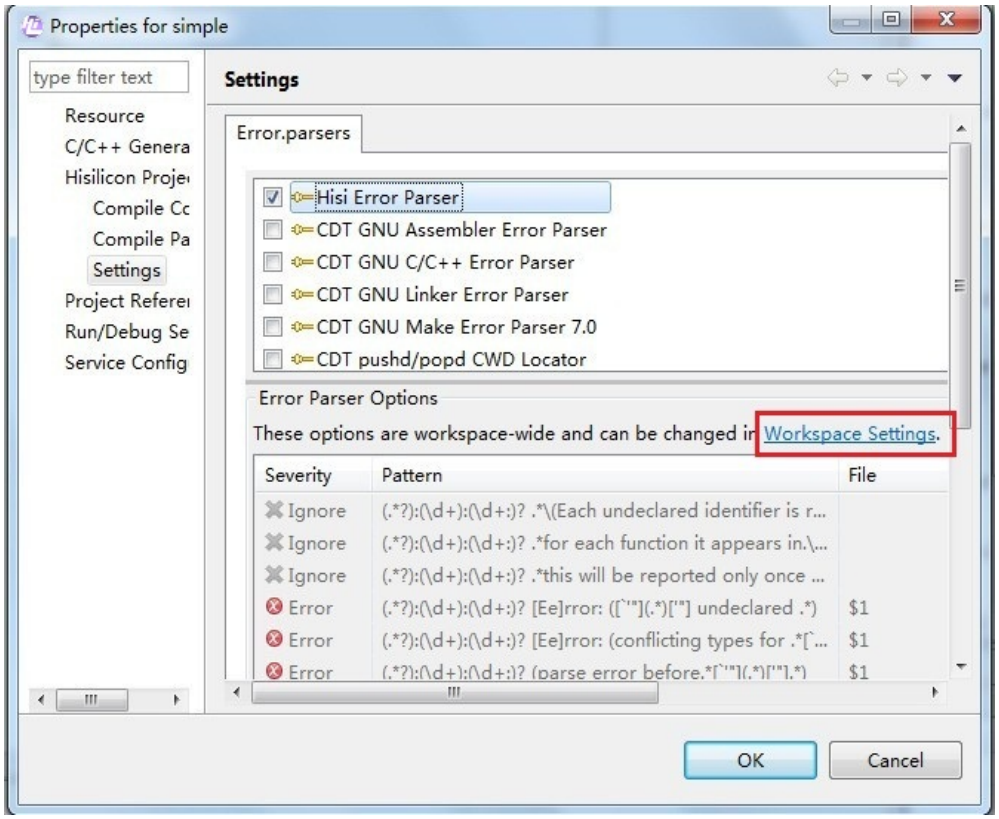


However, not all alarms reported by the tool chain are displayed on the **Problems** tab page. If you want to display the alarm severities on the tab page, you need to set rules of the error parser.

5.7.2 Setting the Error Parser for a Project

Right-click a project name and choose **Properties** from the shortcut menu. Choose **Hisilicon Project** > **Settings**, as shown in Figure 5-54. Select the required parser. To change the parsing rules of the parser, click **Workspace Settings**.

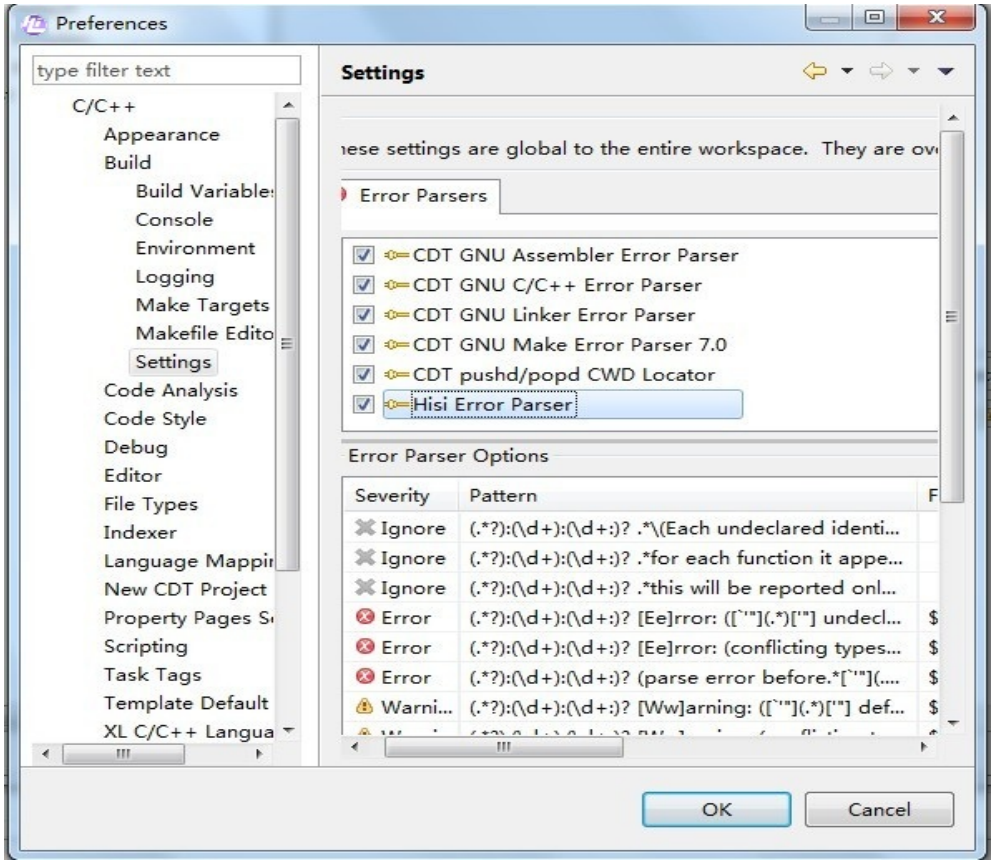
Figure 5-54 Setting the error parser



5.7.3 Setting Rules of Error Parsers

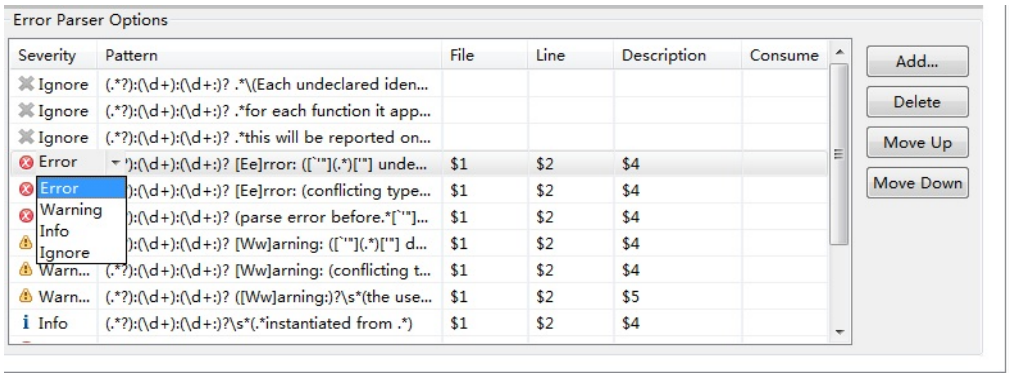
On the toolbar, choose **Windows** > **Preferences** > **C/C++** > **Build** > **Settings**. Then you can add, edit, and remove selected parsers, as shown in Figure 5-55.

Figure 5-55 Preferences



Click the **Severity** column to set the error level, and click the **Pattern** column to enter regular expressions, as shown in Figure 5-56.

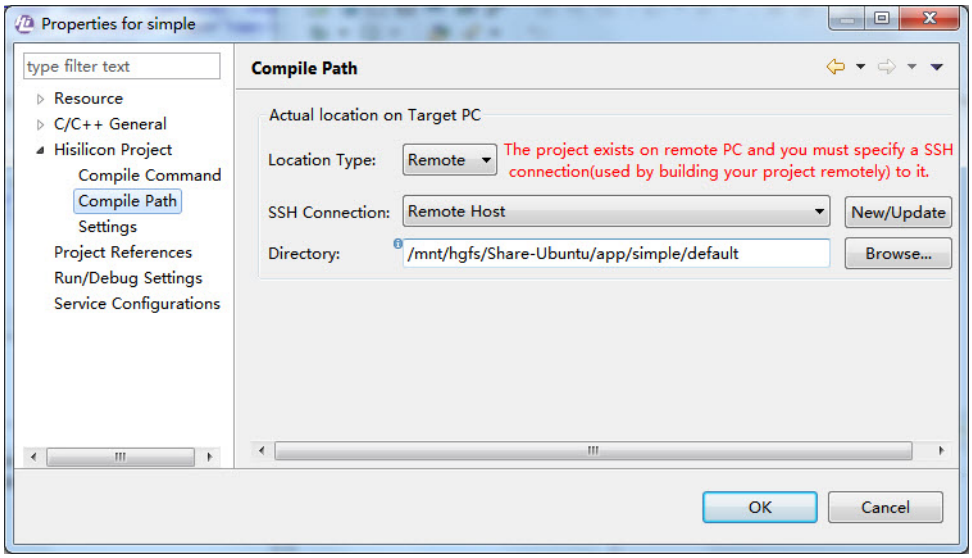
Figure 5-56 Editing error parsers



5.8 Compilation Path Update

Right-click **Project Explorer**, and choose **Properties** from the shortcut menu. The **Properties for simple** dialog box is displayed. Choose **Hisilicon Project** > **Compile Path**, as shown in Figure 5-57. Select an SSH connection from the **SSH Connection** drop-down list, and set the compilation path. If the required SSH connection is not in the **SSH Connection** drop-down list, create an SSH connection. For details, see step 3 in section 3.1.1 "Creating a Remote Project."

Figure 5-57 Compile Path



6 FAQs

6.1 What Should I Pay Attention to During JTAG Debugging?

Problem Description

What should I pay attention to during JTAG debugging?

Solution

Note the following:

- Check whether the Hi-ICE emulator has been prepared.
- Check whether the target board has the JTAG debugging interface.
- Check whether the Hi-ICE is properly connected to the USB port of the PC and the JTAG debugging interface of the target board.
- Check whether the Hi-ICE driver has been successfully installed on the PC.

6.2 Can I Enter Commands in the Debugging Console?

Problem Description

Can I enter commands in the debugging console?

Solution

The debugging console is used only to display information. Therefore, entering commands in the debugging console is not allowed.

6.3 What Are the Connection Types in HiSilicon Projects?

Problem Description

What are the connection types in HiSilicon projects?

Solution

There are three types of connections in HiSilicon projects. For details, see the connection configuration section.

- Compilation network connection: An SSH connection to the compilation server where the source code is stored is required when you create a remote project.
- Board-end network connection: A network connection to the target board is required for debugging Linux applications.
- Board-end JTAG connection: A JTAG connection by using the Hi-ICE emulator is required for debugging bare board programs.

6.4 What Should I Do If the OpenOCD Fails to Be Started?

Problem Description

What should I do if the OpenOCD fails to be started?

Solution

Do as follows:

- Check whether the Hi-ICE driver has been installed.
- Check whether the Hi-ICE device is connected properly.

Figure 6-1 Information indicating that the Hi-ICE driver is not installed or the connection is abnormal

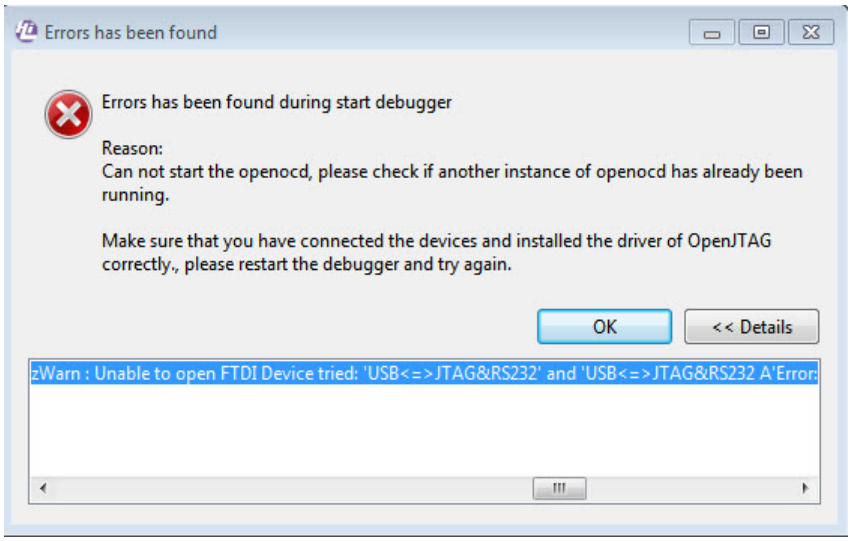


Figure 6-2 shows the information displayed in the console after the OpenOCD is started successfully.

Figure 6-2 Information displayed in the console after the OpenOCD is started successfully

```
Open On-Chip Debugger 0.6.0-rc1
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.sourceforge.net/doc/doxygen/bugs.html
cygwin warning:
  MS-DOS style path detected: G:/runtime-com.hisilicon.cross.ide.product/reference/openocd/interface/openjtag.cfg
  Preferred POSIX equivalent is: /interface/openjtag.cfg
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Info : only one transport option; autoselect 'jtag'
DEPRECATED! use 'adapter_nsrst_delay' not 'jtag_nsrst_delay'
adapter_nsrst_delay: 500
jtag_ntrst_delay: 500
500 kHz
DEPRECATED! use 'adapter_nsrst_delay' not 'jtag_nsrst_delay'
adapter_nsrst_delay: 500
jtag_ntrst_delay: 500
500 kHz
Info : device: 4 "2232C"
Info : deviceID: 341266712
Info : SerialNumber: FTURW4T1A
Info : Description: USB<=>JTAG&RS232 A
Info : clock speed 500 kHz
Info : JTAG tap: hs3716.dap tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
Info : hs3716.dap: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: hs3716.dap tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
reset init...
umap address[0x10000, 0xffffff]
umap address[0x10010000, 0x100fffff]
umap address[0x10140000, 0x1014ffff]
umap address[0x10190000, 0x101dffff]
umap address[0x101fd000, 0x101fdfff]
umap address[0x10205000, 0x10205fff]
umap address[0x10230000, 0x1032ffff]
umap address[0x10460000, 0x167fffff]
umap address[0x16900000, 0x23ffffff]
umap address[0x2a000000, 0x2fffffff]
umap address[0x32000000, 0x33ffffff]
umap address[0x36000000, 0x5fffffff]
umap address[0x60060000, 0x6006ffff]
umap address[0x60090000, 0x600affff]
umap address[0x60160000, 0x17ffffff]
umap address[0x601d0000, 0x7fffffff]
```

6.5 What Do I Do If the Connection Cannot Be Established After the Correct Board IP Address Is Entered During Debugging of an Application?

Problem Description

During debugging of an application, the board IP address is entered correctly, and the system displays a message asking me to enter the user ID and password. However, the telnet connection cannot be established successfully no matter what data is entered.

Cause Analysis

The debugging board uses the Android solution, and the telnet service of the Android solution does not require login information.

Solution

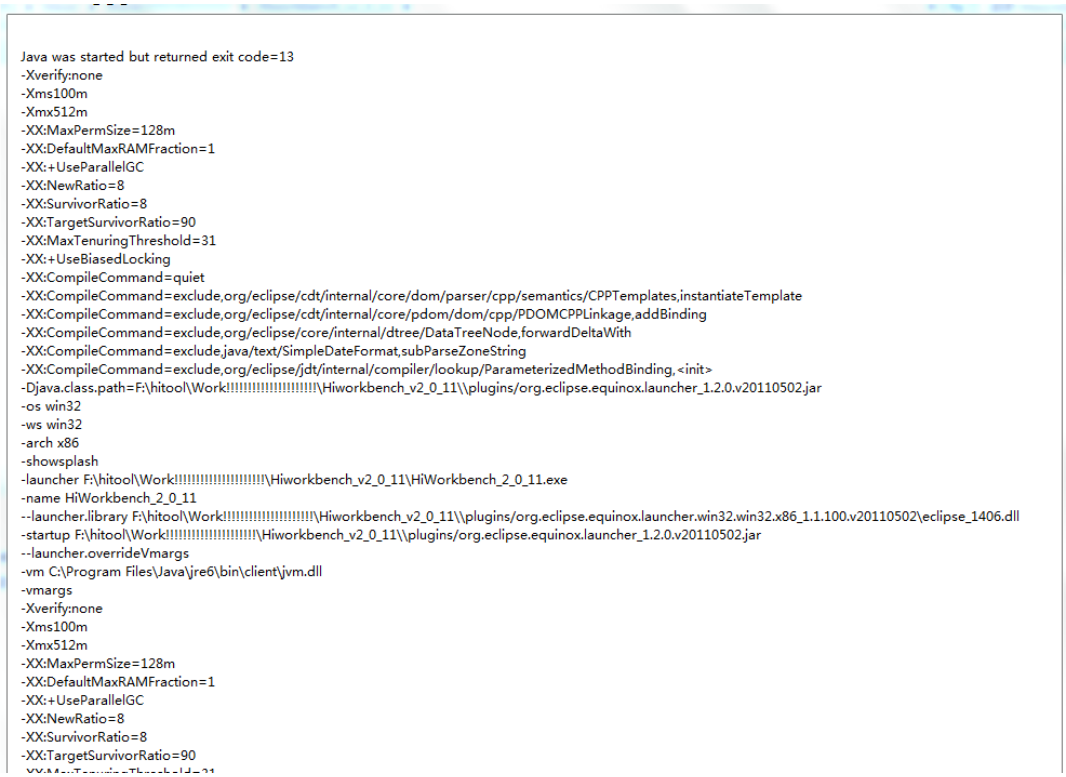
Set **Login.Required** to **false**. For details, see section 3.4.1 "Managing Board Connections."

6.6 What Do I Do If the HiWorkbench Cannot Be Started When It Is Stored in a Path Similar to F:\Work!!!!!!!!!!!!!!!!!!!!\?

Problem Description

When the HiWorkbench is stored in a directory similar to **F:\Work!!!!!!!!!!!!!!!!!!!!**, error information shown in Figure 6-3 is displayed, and the HiWorkbench cannot be started.

Figure 6-3 Path error information



Cause Analysis

The character ! cannot be identified by the Eclipse on which the HiWorkbench is dependent.

Solution

Do not store the HiWorkbench in a path with special characters.

6.7 What Do I Do If an Existing Remote Project Cannot Be Opened After the PC Is Restarted?

Problem Description

After the PC is restarted, a remote project created before cannot be opened in the HiWorkbench.

Cause Analysis

When a remote project is being created, the project path is mapped to the local as a network drive. When the PC is restarted, the mapped network drive is disconnected, and the HiWorkbench cannot find the specified project file.

Solution

Ensure that the network drive is connected and files in the project path can be accessed in the resource manager before opening the project.

A Acronyms and Abbreviations

C	
CDT	C/C++ development toolkit
G	
GDB	GNU Debugger
I	
IDE	integrated drive electronics
IP	Internet Protocol
J	
JDK	Java development kit
JDT	Java development tooling
JRE	Java runtime environment
JTAG	Joint Test Action Group
S	
SSH	secure shell
SVF	serial vector format