Hi3516A/Hi3516D U-boot Porting

# Development Guide

**Issue** 03

**Date** 2015-10-31

HiSilicon Technologies Co., Ltd.

| | |
|---|---|
| Address: | Huawei Industrial Base<br>Bantian, Longgang<br>Shenzhen 518129<br>People's Republic of China |
| Website: | http://www.hisilicon.com |
| Email: | support@hisilicon.com |

# About This Document

## Purpose

This document describes how to port and burn the U-boot (that is, bootloader of the Hi3516A board) on the Hi3516A/Hi3516D board, and how to using ARM debugging tools.

## Related Versions

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

| Product Name | Version |
|---|---|
| Hi3516A | V100 |
| Hi3516D | V100 |

## Intended Audience

This document is intended for:

- Technical support personnel
- Software development engineers

## Change History

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

### Issue 03 (2015-10-31)

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

**Chapter 2 Porting the U-boot**

In section 2.3 and 2.4 are modified.

## Issue 02 (2015-06-16)

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

**Chapter 2 Porting the U-boot**

The direcotry to which **u-boot.bin** and **reg_info.bin** are copied is changed to **osdrv/tools/pc/uboot_tools**.

## Issue 01 (2014-12-20)

This issue is the first official release, which incorporates the following changes:

The contents related to the Hi3516D are added.

## Issue 00B03 (2014-11-10)

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

**Chapter 5 Appendix**

This chapter is added.

## Issue 00B02 (2014-09-14)

This issue is the second draft release.

## Issue 00B01 (2014-07-25)

This issue is the first draft release.

# Contents

# Figures

# Tables

# 1 Overview

## 1.1 Description

The bootloader of the Hi3516A/Hi3516D board uses the U-boot. When the types of the selected peripheral components are different from the types of the components on the board, you need to modify the U-boot configuration file including the information about memory configuration and pin multiplexing.

**NOTE**

This document uses the Hi3516A as an example. Unless otherwise stated, Hi3516D and Hi3516A contents are consistent.

## 1.2 U-boot Directory Structure

Table 1-1 shows the main directory structure of the U-boot. For details, read the **readme** file in the **U-boot** folder.

**Table 1-1** Main directory structure of the U-boot

| Directory | Description |
|---|---|
| arch | Indicates the code of the chip architecture and the entry code of the U-boot. |
| board | Indicates the code of boards, such as the memory driver code. |
| board/hi3516a | Indicates the code of the Hi3516A board. |
| arch/xxx/lib | Indicates the code of architecture, such as the common code of ARM and microprocessor without interlocked pipeline stages (MIPS) architecture. |
| include | Indicates header files. |
| include/configs | Indicates the configuration files of boards. |
| common | Indicates the implementation files of functions or commands. |
| drivers | Indicates the driver code of Ethernet ports, flash memories, and serial ports. |

| Directory | Description |
|---|---|
| net | Indicates the implementation files of network protocols. |
| fs | Indicates the implementation files of file systems. |

# 2 Porting the U-boot

## 2.1 U-boot Hardware Environment

Peripheral components on the Hi3516A demo board include the DDR SDRAM, NAND flash, SPI flash, and SPI NAND flash. For details, see the *Hi3516A Component Compatibility List*.

## 2.2 Compiling the U-boot

After preceding operations are complete, you can compile the U-boot by running the following commands:

**Step 1** Configure the compilation environment.

```
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- hi3516a_config
or
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- hi3516a_spinand_config
```

If the compilation is successful, the **u-boot.bin** file is generated in the **U-boot** folder.

**NOTE**

**hi3516a_config** applies to the SPI flash and NAND flash, whereas **hi3516a_spinand_config** applies to the SPI NAND flash.

**Step 2** Compile the U-boot.

```
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux-
```

**NOTE**

The **CROSS_COMPILE** parameter indicates the tool chain. In this document, the **CROSS_COMPILE = arm-hisiXXX-linux-** parameter indicates the following two situations:

- Hi3516A_V100R001C01SPCxxx corresponds to uclibc. If the uclibc tool chain is used, the **CROSS_COMPILE** parameter is set to **arm-hisiv300-linux-**.
- Hi3516A_V100R001C02SPCxxx corresponds to glibc. If the glibc tool chain is used, the **CROSS_COMPILE** parameter is set to **arm-hisiv400-linux-**.

**CAUTION**

The **u-boot.bin** file is not the final u-boot image.

**----End**

## 2.3 Configuring the DDR

On Windows, open the configuration table in **osdrv/ tools/pc/uboot_tools** of the SDK. If different DDR SDRAMs are used, modify the **mddrc_dmc1**, **mddrc_dmc2**, and **mddrc_phy** sheets in the configuration table based on DDR features.

## 2.4 Configuring Pin Multiplexing

If the pin multiplexing relationship changes, modify the **muxctrl_reg** sheet in the configuration table.

## 2.5 Generating the Final U-boot Image

Perform the following steps:

**Step 1** Save settings after modifying the configuration sheet.

**Step 2** Click **Generage reg bin file** on the first tab page of the configuration sheet to generate the temporary file **reg_info.bin**.

**Step 3** Copy **u-boot.bin** (it is generated after the U-boot is compiled) and **reg_info.bin** to **osdrv/tools/pc/uboot_tools** of the SDK, and run the following command:

```
mkboot.sh reg_info_hi3516a.bin u-boot-hi3516a.bin
```

**u-boot-hi3516a.bin** is the final u-boot image that can run on the board.

**----End**

# 3 Burning the U-boot

## 3.1 Overview

If the U-boot has run on the board to be ported, you can update the U-boot by connecting the board to the server over the serial port or Ethernet port.

If the U-boot is burnt for the first time, burn it by using the fastboot or DS-5 tool over the Ethernet port. According to chip features, you must initialize the DDR and chip by running the scripts provided in the Hi3516A SDK before using the DS-5 tool. When different DDRs are used, the initialization scripts can be used only when they are reconfigured.

## 3.2 Burning the U-boot by Using the BOOTROM

For details, see the *Fastboot Burning Tool Application Notes*

## 3.3 Burning the U-boot to Two Types of Flash Memories

### 3.3.1 Burning the U-boot to the SPI Flash

Perform the following steps:

**Step 1** Run the following commands in the HyperTerminal after the U-boot runs in the memory:

```
hisilicon# mw.b 0x82000000 ff 0x100000     /*Set the DDR value of all Fs.*/
hisilicon# tftp 0x82000000 u-boot-hi3516a.bin /*Download the U-boot to the
DDR.*/
hisilicon# sf probe 0                   /*Detect and initialize the SPI
flash.*/
hisilicon# sf erase 0x0 0x100000           /*Erase 1 MB capacity of the SPI
flash.*/
hisilicon# sf write 0x82000000 0x0 0x100000   /*Write the U-boot from the DDR
to the SPI flash.*/
```

**Step 2** Restart the system. The U-boot is burnt successfully.

**----End**

**CAUTION**

In the current version, the blocks of the SPI flash can be locked by running **sf lock**. If the blocks are locked, running the erase and write commands in earlier versions (no error displayed) or running **sf erase** and **sf write** commands in the current version has no effect. In this case, the erase and write operations can be performed only after the blocks of the SPI flash are unlocked by running **sf lock**. For details, see section 5.1.1 "Block Protection Command for the SPI Flash."

## 3.3.2 Burning the U-boot to the NAND Flash

Perform the following steps:

**Step 1** Run the following commands in the HyperTerminal after the U-boot runs in the memory:

```
hisilicon# nand erase 0 100000            /*Erase 1 MB capacity of the NAND
flash.*/
hisilicon# mw.b 0x82000000 ff 100000       /*Set the DDR value to all Fs.*/
hisilicon# tftp 0x82000000 u-boot-hi3516a.bin    /*Download the U-boot to the
DDR.*/
hisilicon# nand write 0x82000000 0 100000   /*Write the U-boot from the DDR
to the NAND flash.*/
```

**Step 2** Restart the system. Then the U-boot is burnt successfully.

**----End**

**CAUTION**

The same NAND command is used to burn the NAND flash and SPI NAND flash. Therefore, the NAND flash and SPI NAND flash cannot be connected on the board at the same time.

# 4 Using ARM Debugging Tools

## 4.1 Overview

The ARM Development Studio 5 (DS-5) is an end-to-end software development toolkit used on the Linux and Android platforms. It covers the development of the startup code, kernel porting, application, and bare board debugging. The DS-5 provides the kernel space debugger and the application with the tracking function, system-wide performance analyzer, real-time system simulator, and compiler. These functions are included in the customized, powerful, and friendly Eclipse-based integrated development environment (IDE). The DS-5 can help engineers to develop and optimize systems based on Linux for platforms supported by ARM, shorten the development and test cycles, and develop highly efficient software.

The DS-5 includes:

- DS-5 Eclipse: An IDE which integrates the compilation tools with the debugging tools.
- DS-5 Debug
- Real-time system models （RTSM）
- ARM pipeline performance analyzer

This chapter describes how to use the following tools to debug the ARM processor:

- DS-5 Eclipse
- DS-5 Debug

## 4.2 ARM Debugging Tools

### 4.2.1 DS-5 Eclipse

The DS-5 Eclipse is an Eclipse-based IDE. It integrates the compilation tools and debugging tools of ARM, and ARM Linux GNU tool chain for ARM Linux target board development. The DS-5 Eclipse provides project management, editor, and view.

### 4.2.2 DS-5 Debug

The DS-5 Debug is a graphical debugger which supports software development debugging on the ARM target board and RTSM. Because of its comprehensive and direct view, you can easily debug Linux and bare board programs, including source program synchronization and disassembling, stack calling management, operations of the memory, register, expression, variable, thread, and breakpoint, as well as code tracking.

The DS-5 Debug management window can be used to execute source codes or instructions step by step, and view the latest data in other views after code running. You can also set breakpoints or checkpoints to pause the application to learn the status after application execution. The tracking view can be used on some target boards to track function execution in the application in the sequence of program running.

## 4.3 Using ARM Debugging Tools

To use the DS-5 to debug the program or burn the U-boot to the development board, you must create a configuration database for the target platform, and then connect the DS-5 to the target platform.

For details about how to use ARM debugging tools, see the documents provided by ARM. To use the DS-5, perform the following steps:

**Step 1** Install the DS-5.

**Step 2** Create a configuration database for the target platform.

1. Run **Debug Hardware Configure** to generate the configuration file of the chip.
2. Use this configuration file to generate the configuration database for the target platform.
3. Add the configuration database for the target platform to the system.

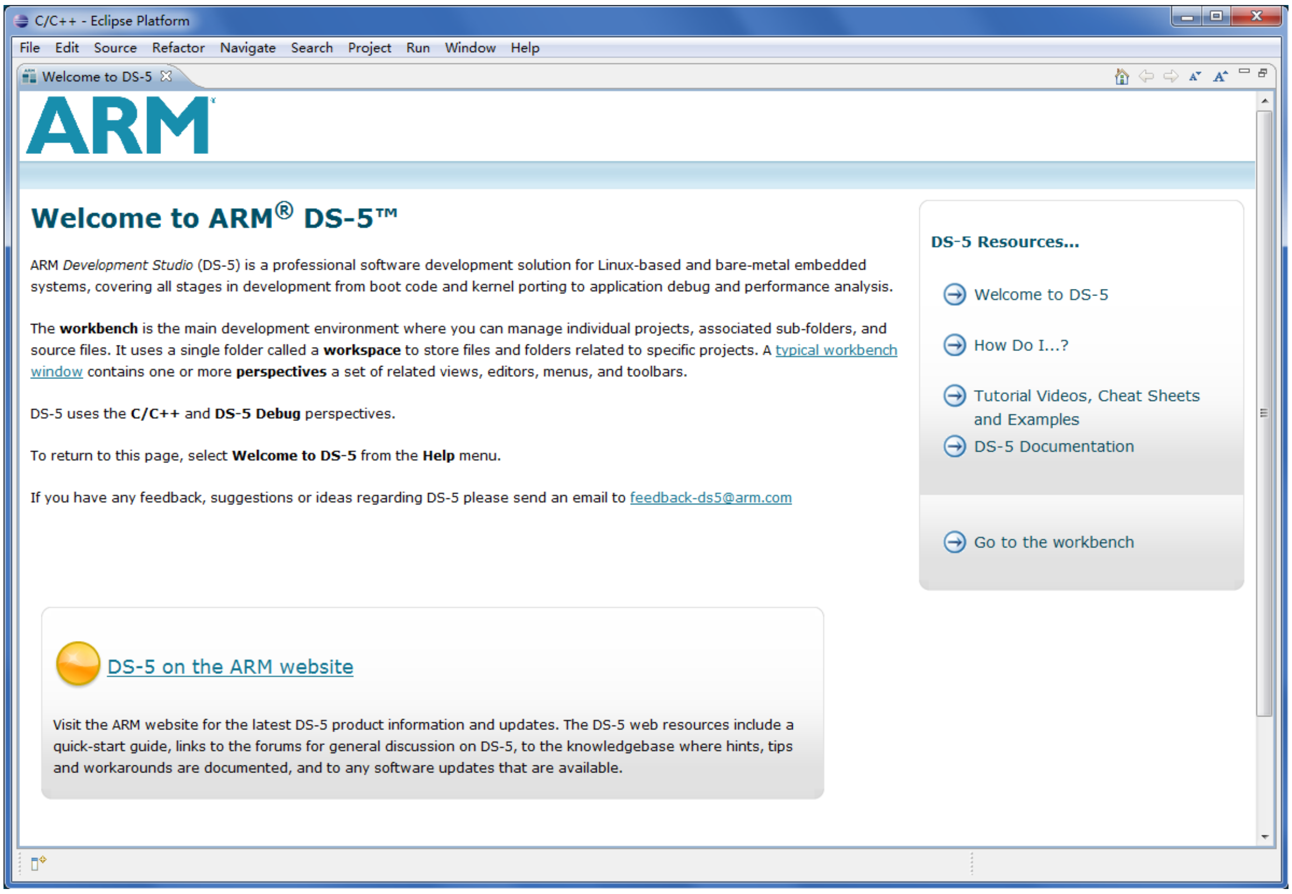
**Step 3** Create a new connection to connect the DS-5 to the target platform by using the configuration database for the target platform.

**----End**

### 4.3.1 Installing the DS-5

The DS-5 is the setup program of the DS-5 Eclipse provided by ARM. Before installation, read relevant documents provided by ARM. After installation, start the DS-5 Eclipse. See Figure 4-1.

**Figure 4-1** Startup GUI of the DS-5 Eclipse


C/C++ - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome to DS-5
ARM

Welcome to ARM® DS-5™

ARM *Development Studio* (DS-5) is a professional software development solution for Linux-based and bare-metal embedded systems, covering all stages in development from boot code and kernel porting to application debug and performance analysis.

The **workbench** is the main development environment where you can manage individual projects, associated sub-folders, and source files. It uses a single folder called a **workspace** to store files and folders related to specific projects. A typical workbench window contains one or more **perspectives** a set of related views, editors, menus, and toolbars.

DS-5 uses the **C/C++** and **DS-5 Debug** perspectives.

To return to this page, select **Welcome to DS-5** from the **Help** menu.

If you have any feedback, suggestions or ideas regarding DS-5 please send an email to feedback-ds5@arm.com

DS-5 on the ARM website

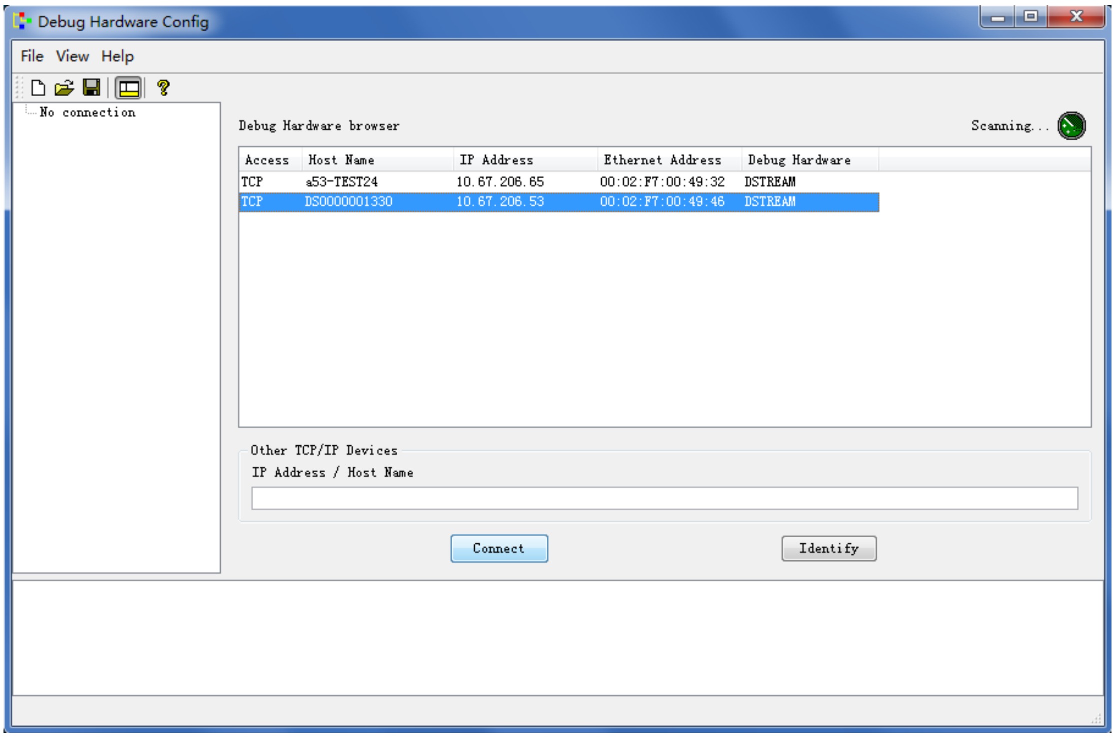
Visit the ARM website for the latest DS-5 product information and updates. The DS-5 web resources include a quick-start guide, links to the forums for general discussion on DS-5, to the knowledgebase where hints, tips and workarounds are documented, and to any software updates that are available.

## 4.3.2 Creating a Configuration Database for the Target Platform

Perform the following steps:

**Step 1** Choose **Start** > **All Programs** > **ARM DS-5** > **Debug Hardware** > **Debug Hardware Configure**, and run **Debug Hardware Configure** to scan connected simulators. Select the specified simulator and click **Connect**. See Figure 4-2.

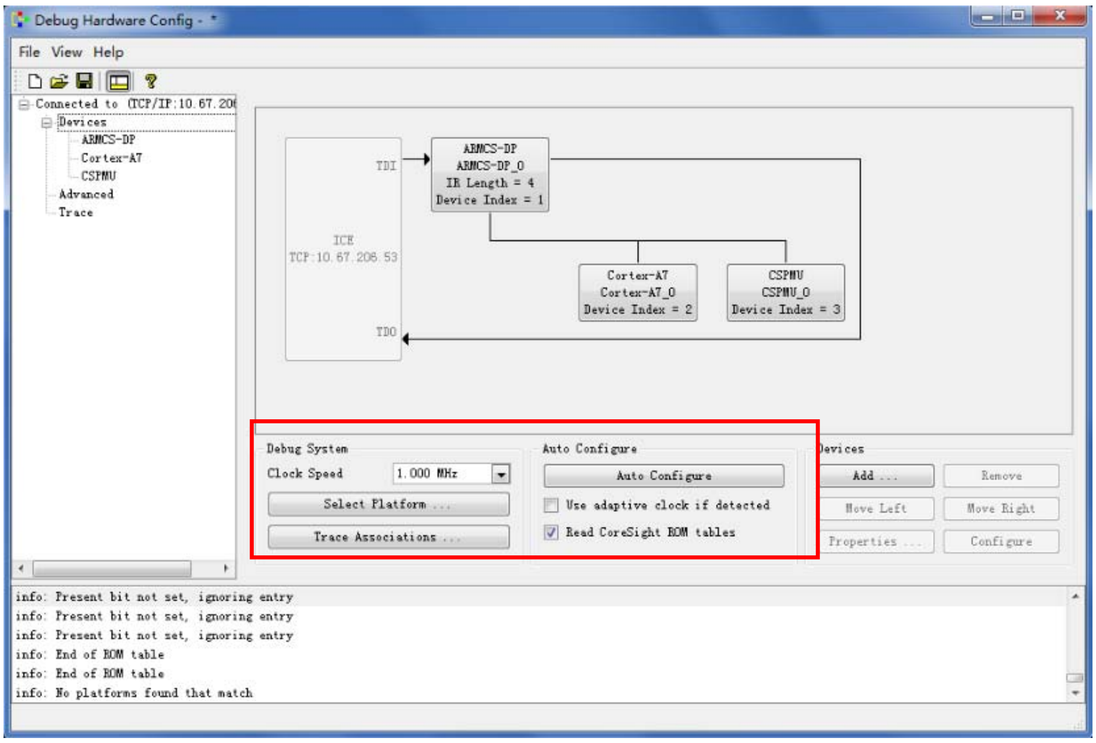
**Figure 4-2** Device scanning window



| Access | Host Name | IP Address | Ethernet Address | Debug Hardware |
|---|---|---|---|---|
| TCP | a53-TEST24 | 10.67.206.65 | 00:02:F7:00:49:32 | DSTREAM |
| TCP | DS0000001330 | 10.67.206.53 | 00:02:F7:00:49:46 | DSTREAM |

**Step 2** In the displayed window, click **Auto Configure** to automatically configure the chip. See Figure 4-3.

**Figure 4-3** Chip configuration window



Clock Speed 1.000 MHz

Auto Configure

[ ] Use adaptive clock if detected

[✓] Read CoreSight ROM tables

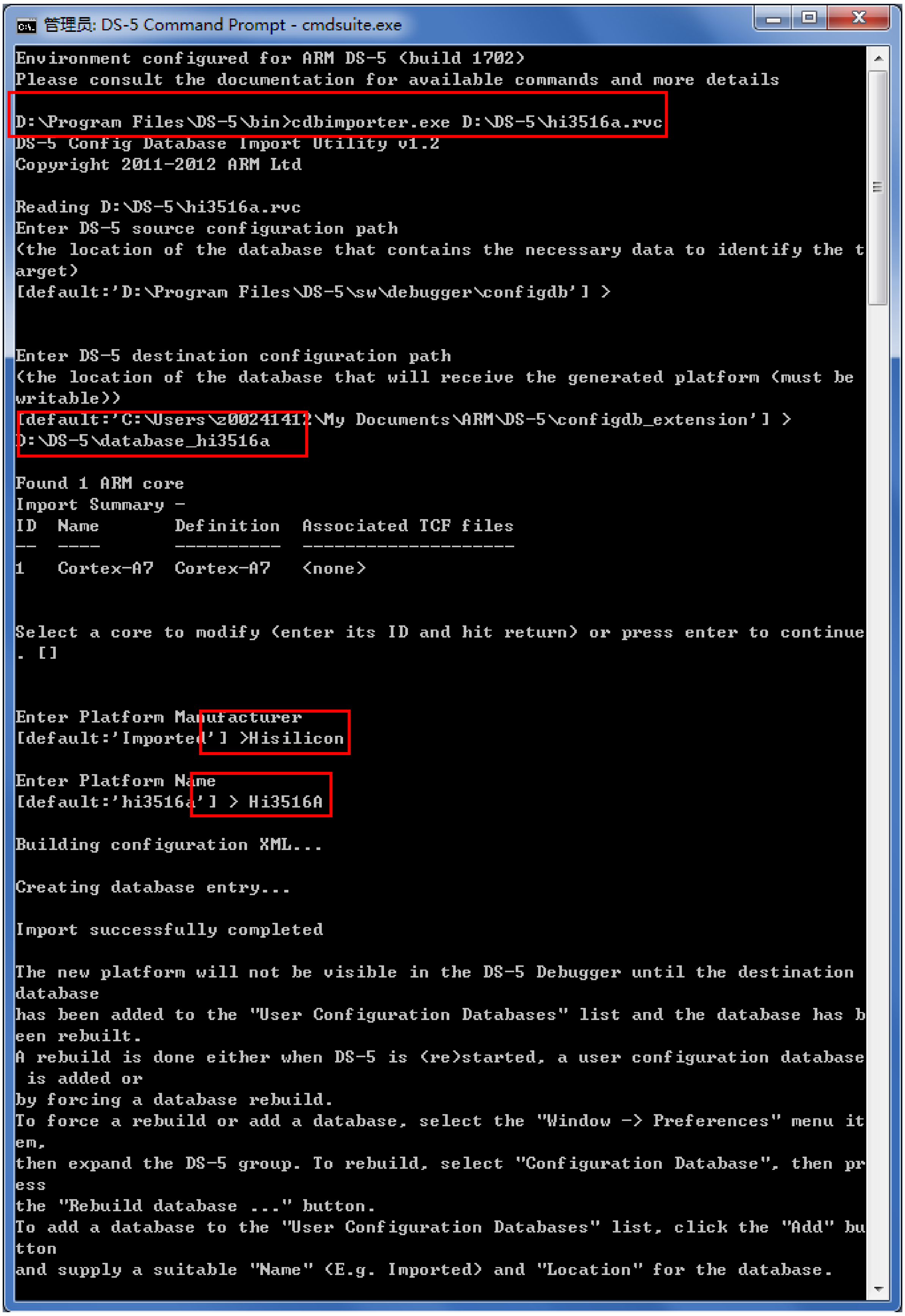
```
info: Present bit not set, ignoring entry
info: Present bit not set, ignoring entry
info: Present bit not set, ignoring entry
info: End of ROM table
info: End of ROM table
info: No platforms found that match
```

**Step 3** Exit and save the configuration file to the specified path such as **D:\DS-5\hi3516a.rvc**.

**Step 4** Generate the configuration database for the target platform. See Figure 4-4.

1. Run **cdbimporter.exe** to read the hi3516a.rvc configuration file of the chip.
2. Specify the source database path that can identify the target platform and press **Enter**. The default path is used in this example.
3. Specify the path for saving the configuration database for the target platform such as **D:\DS-5\database_hi3516a**.
4. Specify the platform manufacturer such as **Hisilicon**.
5. Specify the platform name such as **Hi3516A**.

**Figure 4-4** Window for the command prompts of the DS-5



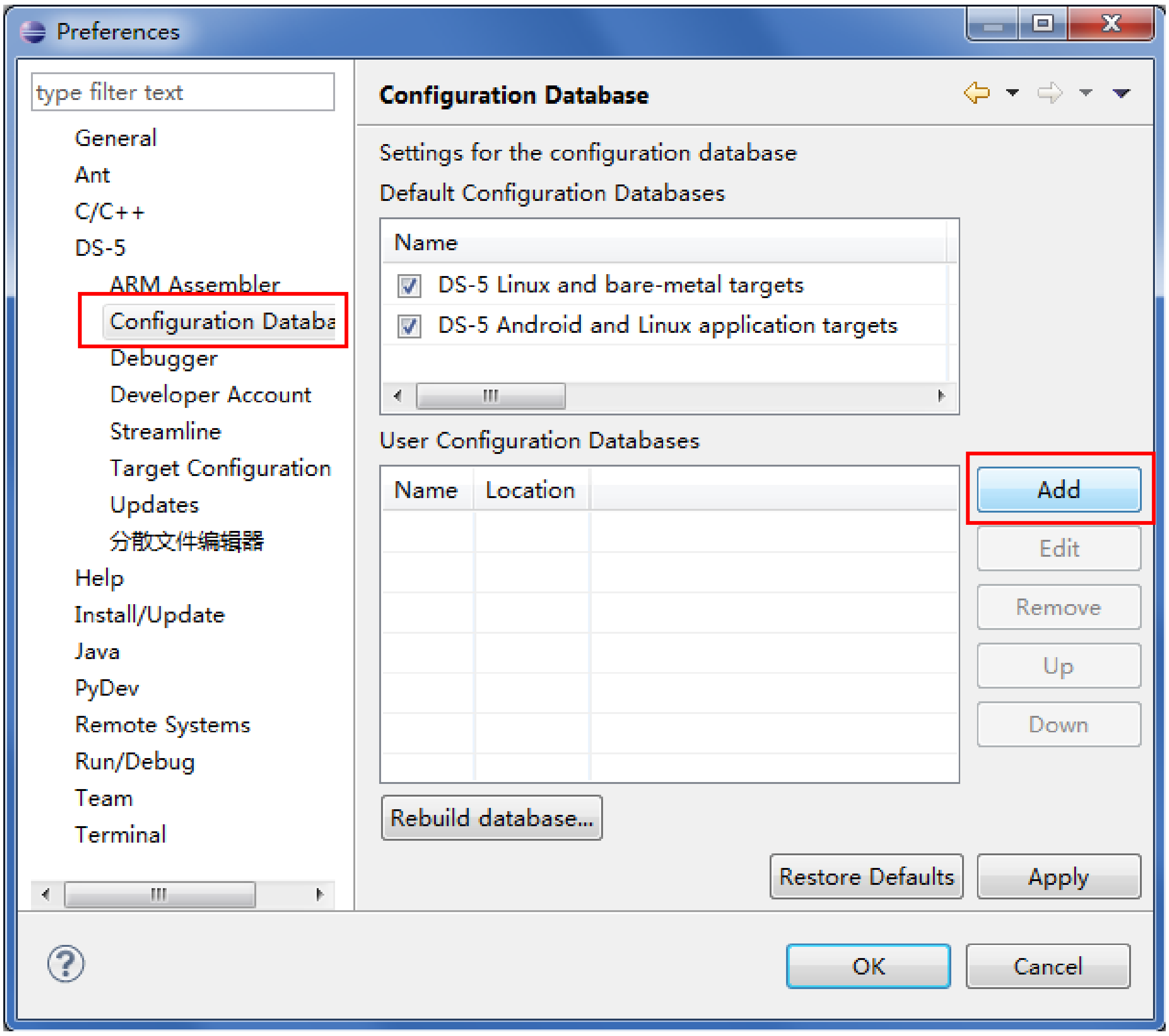
管理员: DS-5 Command Prompt - cmdsuite.exe

```
Environment configured for ARM DS-5 (build 1702)
Please consult the documentation for available commands and more details

D:\Program Files\DS-5\bin>cdbimporter.exe D:\DS-5\hi3516a.rvc
DS-5 Config Database Import Utility v1.2
Copyright 2011-2012 ARM Ltd

Reading D:\DS-5\hi3516a.rvc
Enter DS-5 source configuration path
(the location of the database that contains the necessary data to identify the t
arget)
[default:'D:\Program Files\DS-5\sw\debugger\configdb'] >


Enter DS-5 destination configuration path
(the location of the database that will receive the generated platform (must be
writable))
[default:'C:\Users\z00241412\My Documents\ARM\DS-5\configdb_extension'] >
D:\DS-5\database_hi3516a

Found 1 ARM core
Import Summary -
ID   Name       Definition  Associated TCF files
--   ----       ----------  --------------------
1    Cortex-A7  Cortex-A7   <none>


Select a core to modify (enter its ID and hit return) or press enter to continue
. []


Enter Platform Manufacturer
[default:'Imported'] >Hisilicon

Enter Platform Name
[default:'hi3516a'] > Hi3516A

Building configuration XML...

Creating database entry...

Import successfully completed

The new platform will not be visible in the DS-5 Debugger until the destination
database
has been added to the "User Configuration Databases" list and the database has b
een rebuilt.
A rebuild is done either when DS-5 is (re)started, a user configuration database
 is added or
by forcing a database rebuild.
To force a rebuild or add a database, select the "Window -> Preferences" menu it
em,
then expand the DS-5 group. To rebuild, select "Configuration Database", then pr
ess
the "Rebuild database ..." button.
To add a database to the "User Configuration Databases" list, click the "Add" bu
tton
and supply a suitable "Name" (E.g. Imported) and "Location" for the database.
```

**Step 5** Add the configuration database for the target platform to the system.

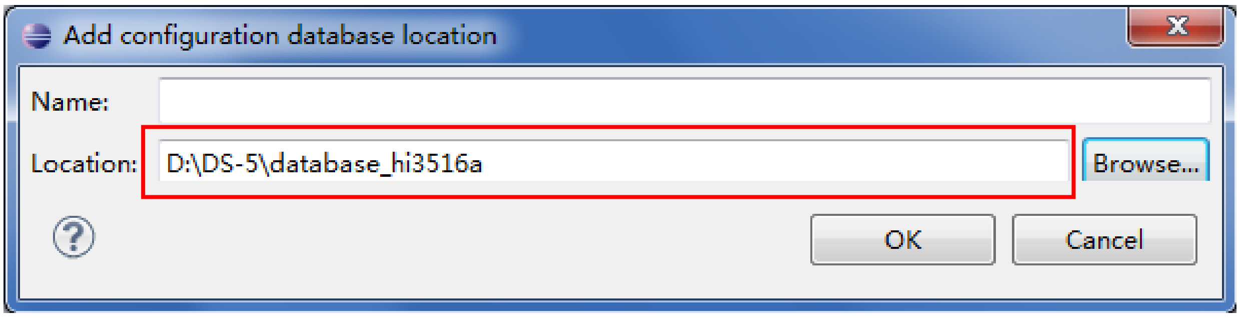
1. Choose **Windows** > **Preferences** from the main menu. In the configuration tree, choose **DS-5** > **Configuration Database**. See Figure 4-5.

**Figure 4-5** Preferences window



2. Click **add**, specify the path for saving the configuration database for the target platform such as **D:\DS-5\database_hi3516a** in the displayed **Add Configuration Database** dialog box. See Figure 4-6.

**Figure 4-6** Add configuration Database dialog box



Location: D:\DS-5\database_hi3516a

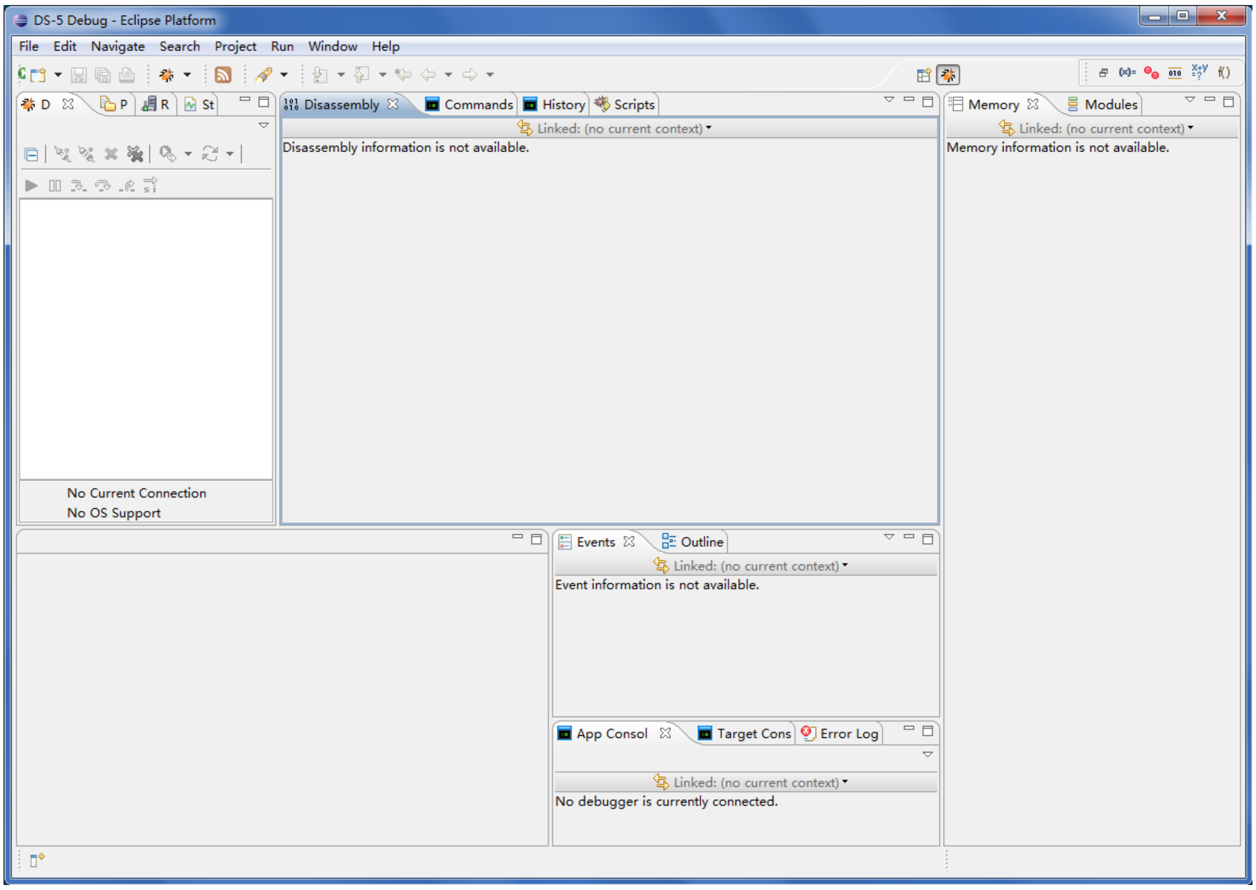
**----End**

## 4.3.3 Connecting the DS-5 to the Target Platform

Perform the following steps:

**Step 1** Choose **Windows** > **Open Perspective** > **DS-5 Debug** from the main menu. See Figure 4-7.

**Figure 4-7** DS-5 Debug window



Linked: (no current context)

Disassembly information is not available.

Memory information is not available.

No Current Connection
No OS Support

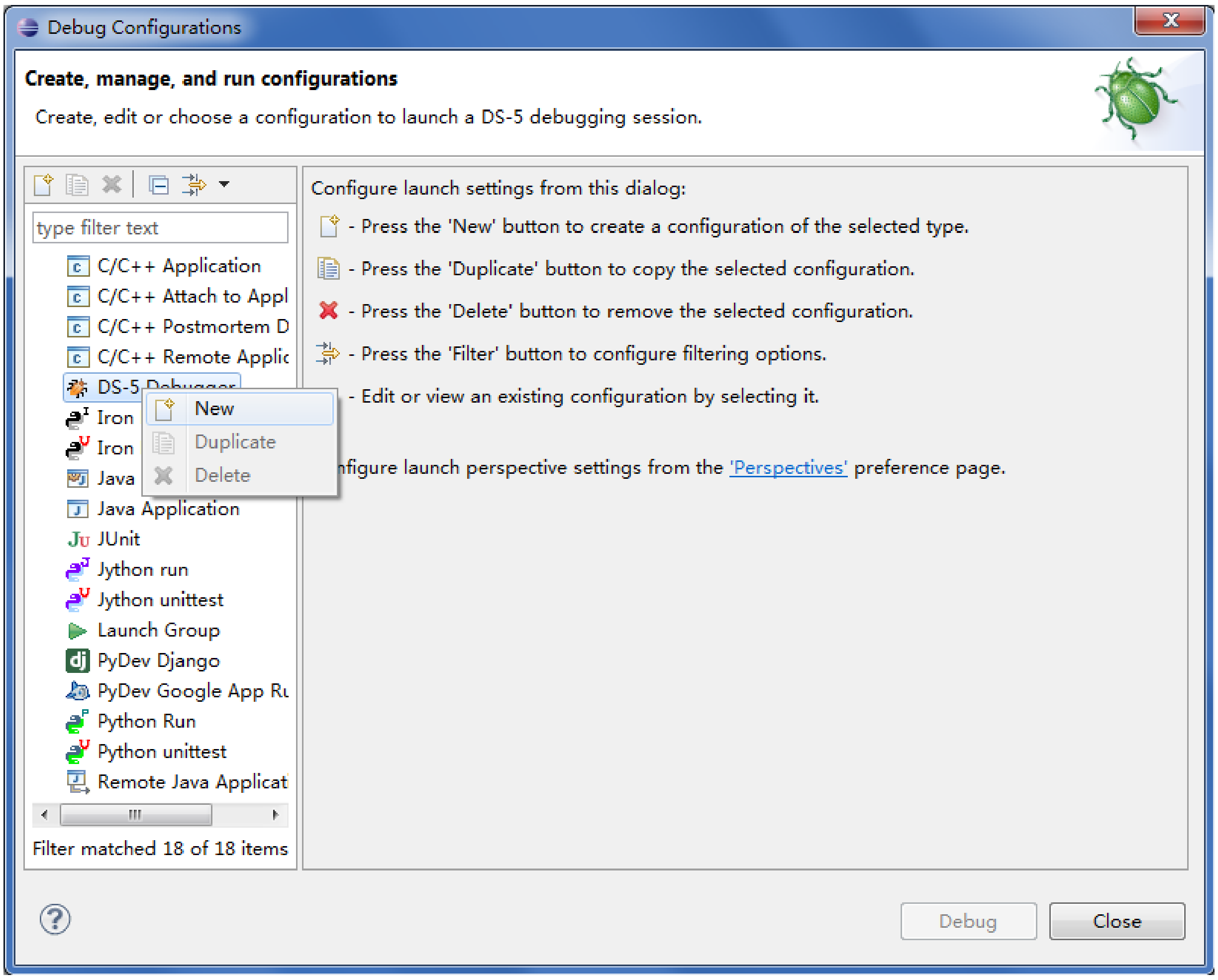
Event information is not available.

No debugger is currently connected.

**Step 2** Choose **Debug Configure** from the **Run** menu, and open the **Debug Configure** window.

**Step 3** Right-click the **DS-5 Debugger** configuration tree, and click **New** to create a new configuration in the displayed menu. See Figure 4-8.

**Figure 4-8** Debug Configure window



Debug Configurations

**Create, manage, and run configurations**

Create, edit or choose a configuration to launch a DS-5 debugging session.

type filter text

- C/C++ Application
- C/C++ Attach to Appl
- C/C++ Postmortem D
- C/C++ Remote Applic
- DS-5 Debugger
- Iron
- Iron
- Java
- Java Application
- JUnit
- Jython run
- Jython unittest
- Launch Group
- PyDev Django
- PyDev Google App Ru
- Python Run
- Python unittest
- Remote Java Applicat

Filter matched 18 of 18 items

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

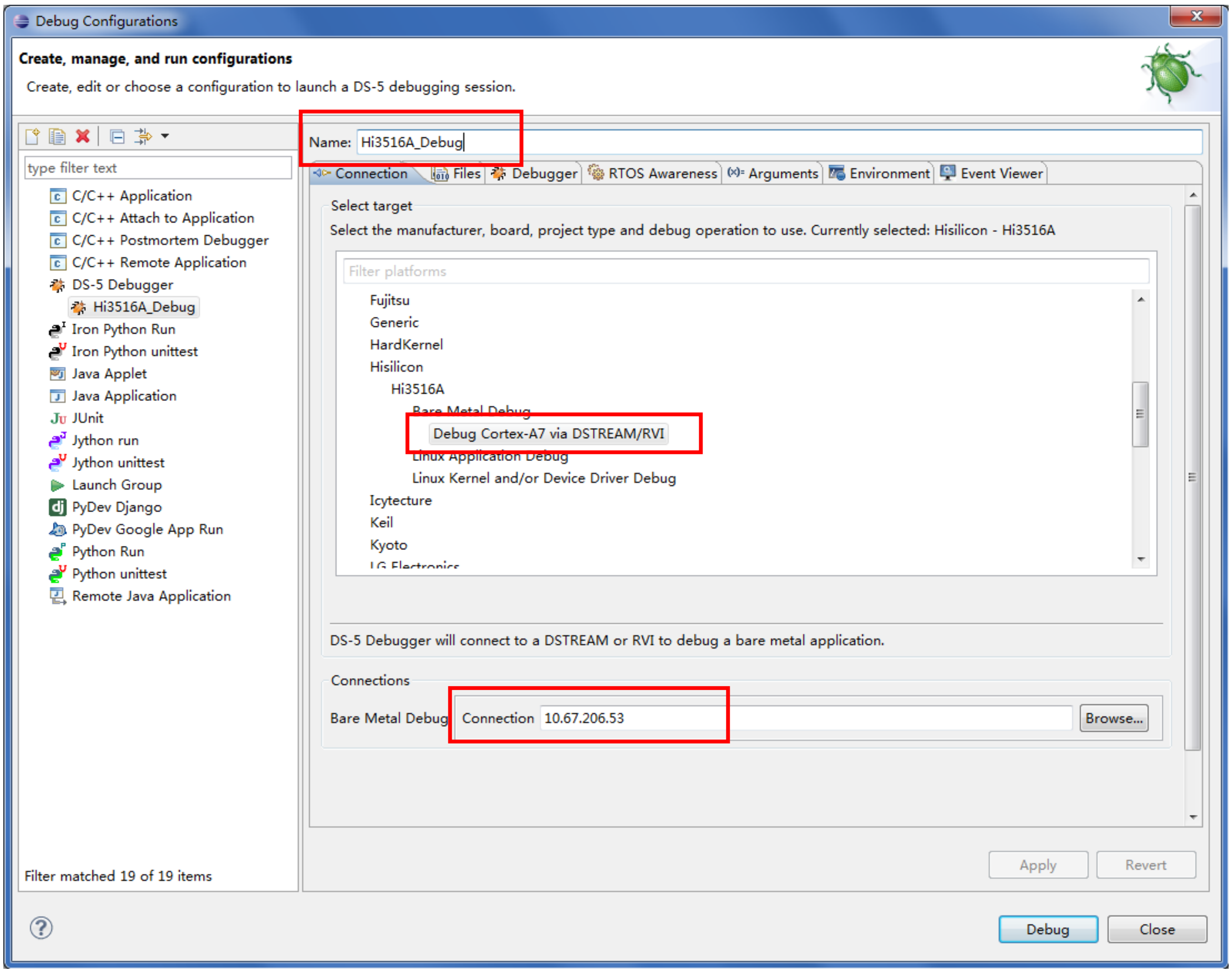
nfigure launch perspective settings from the 'Perspectives' preference page.

**Step 4** Enter a name for the new configuration in the **Name** text box such as **Hi3516A_Debug**.

**Step 5** Click the **Connection** tab page to configure a target connection for the DS-5 Debug.

**Step 6** Choose **Hisilicon** > **Hi3516A** > **Bare Metal Debug** > **Debug Cortex-A7 via DSTREAM/RVI** to select the added configuration database for the target platform, and enter the IP address of the DS-5 in the text box. See Figure 4-9.

**Figure 4-9** Debug Configure window



Debug Configurations

**Create, manage, and run configurations**

Create, edit or choose a configuration to launch a DS-5 debugging session.

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- DS-5 Debugger
  - Hi3516A_Debug
- Iron Python Run
- Iron Python unittest
- Java Applet
- Java Application
- JUnit
- Jython run
- Jython unittest
- Launch Group
- PyDev Django
- PyDev Google App Run
- Python Run
- Python unittest
- Remote Java Application

Filter matched 19 of 19 items

Name: Hi3516A_Debug

Select target

Select the manufacturer, board, project type and debug operation to use. Currently selected: Hisilicon - Hi3516A

Filter platforms

- Fujitsu
- Generic
- HardKernel
- Hisilicon
  - Hi3516A
    - Bare Metal Debug
      - Debug Cortex-A7 via DSTREAM/RVI
    - Linux Application Debug
    - Linux Kernel and/or Device Driver Debug
- Icytecture
- Keil
- Kyoto
- LG Electronics

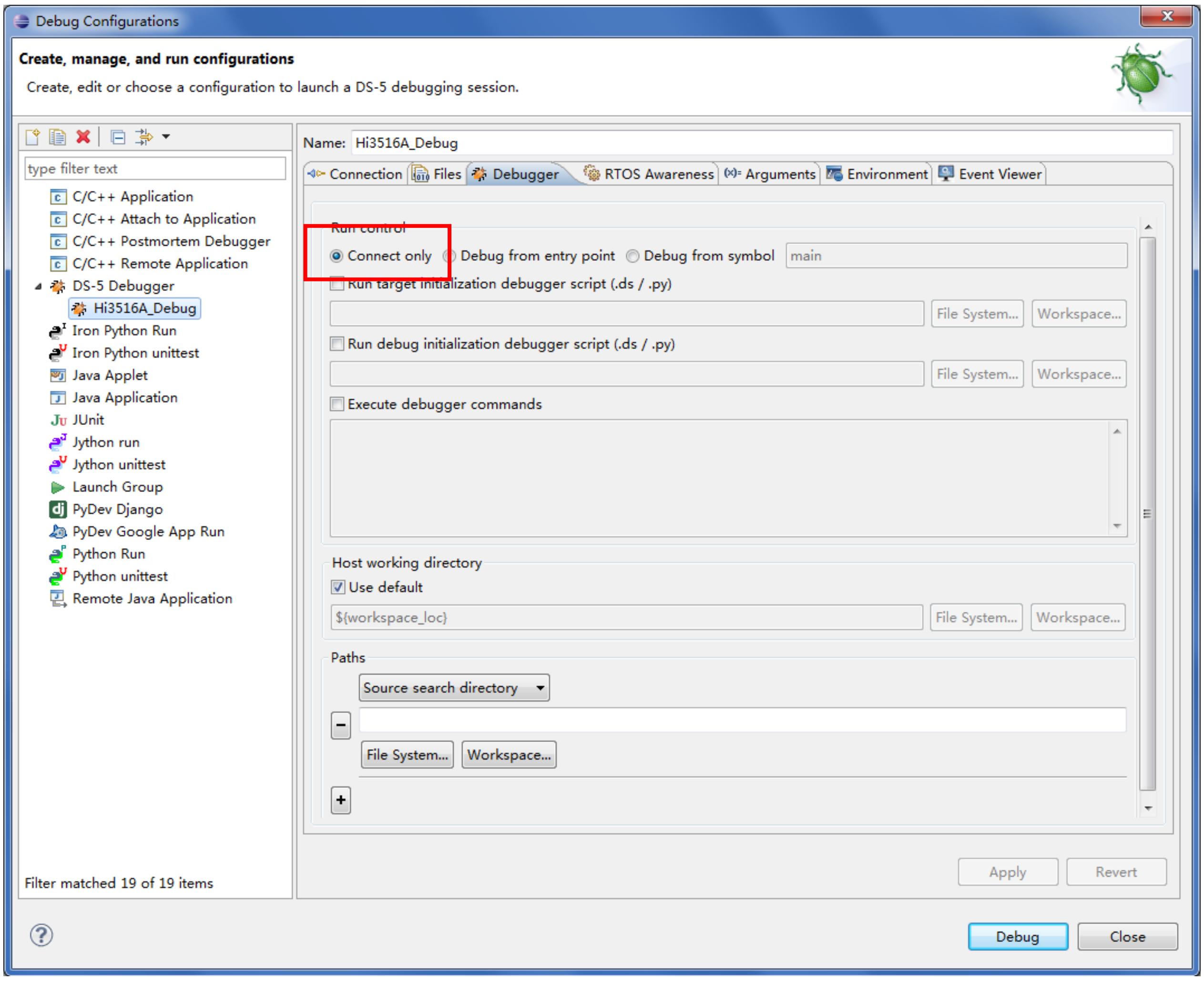
DS-5 Debugger will connect to a DSTREAM or RVI to debug a bare metal application.

Connections

Bare Metal Debug Connection 10.67.206.53

**Step 7** Select **Connect Only** on the **Debugger** tab page. See Figure 4-10.

**Figure 4-10** Debug Configure window



Debug Configurations

**Create, manage, and run configurations**

Create, edit or choose a configuration to launch a DS-5 debugging session.

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- DS-5 Debugger
  - Hi3516A_Debug
- Iron Python Run
- Iron Python unittest
- Java Applet
- Java Application
- JUnit
- Jython run
- Jython unittest
- Launch Group
- PyDev Django
- PyDev Google App Run
- Python Run
- Python unittest
- Remote Java Application

Filter matched 19 of 19 items

Name: Hi3516A_Debug

Connection | Files | Debugger | RTOS Awareness | Arguments | Environment | Event Viewer

Run control

[✓ Connect only] Debug from entry point  Debug from symbol  main

☐ Run target initialization debugger script (.ds / .py)  File System... Workspace...

☐ Run debug initialization debugger script (.ds / .py)  File System... Workspace...

☐ Execute debugger commands

Host working directory

☑ Use default

${workspace_loc}  File System... Workspace...

Paths

Source search directory  File System... Workspace...

Apply  Revert

Debug  Close

**Step 8** Click **Debug** to connect the DS-5 to the target platform.

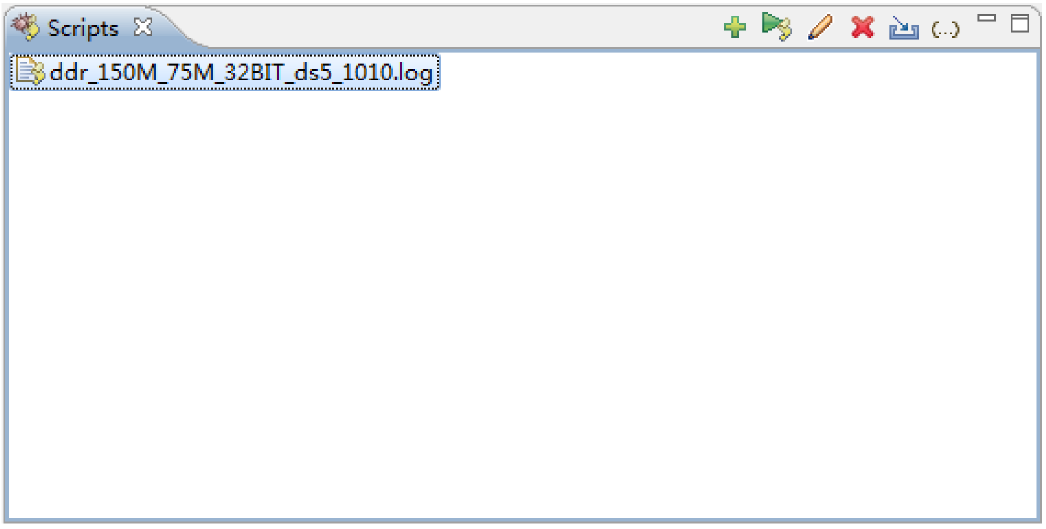
**----End**

## 4.4 Burning Images to a Flash Memory by Using the Simulator

### 4.4.1 Initializing the Memory

In the **Scripts** window, click to import the memory initialization script, and click to run the memory initialization script (if the simulator is running, click in the **Debug Control** window to stop it). See Figure 4-11.

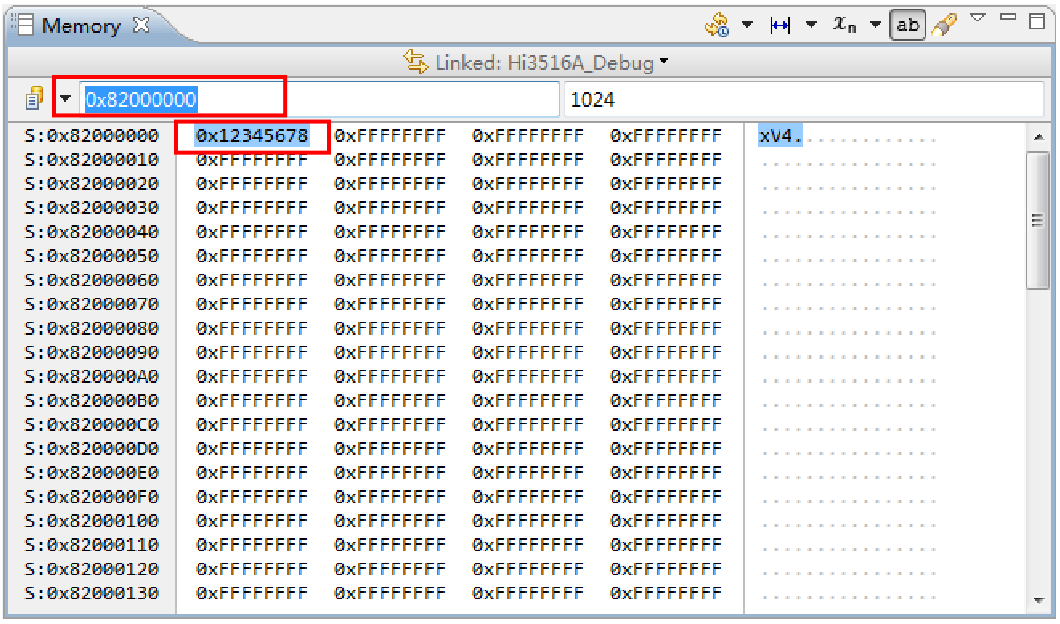
**Figure 4-11** Scripts window



Use the following method to check whether the memory is successfully initialized.

Enter the memory address such as **0x82000000** in the **Memory** window. Press **Enter** to view whether the memory value is displayed in the window shown in Figure 4-12. If values are displayed and can be changed, the memory is successfully initialized. To change a memory value such as **0x82000000**, double-click the value, enter a new value such as **0x12345678**, and press **Enter**. See Figure 4-12.

**Figure 4-12** Memory window



```
S:0x82000000   0x12345678  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   xV4.............
S:0x82000010   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000020   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000030   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000040   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000050   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000060   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000070   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000080   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000090   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x820000A0   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x820000B0   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x820000C0   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x820000D0   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x820000E0   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x820000F0   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000100   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000110   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000120   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
S:0x82000130   0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF  0xFFFFFFFF   ................
```

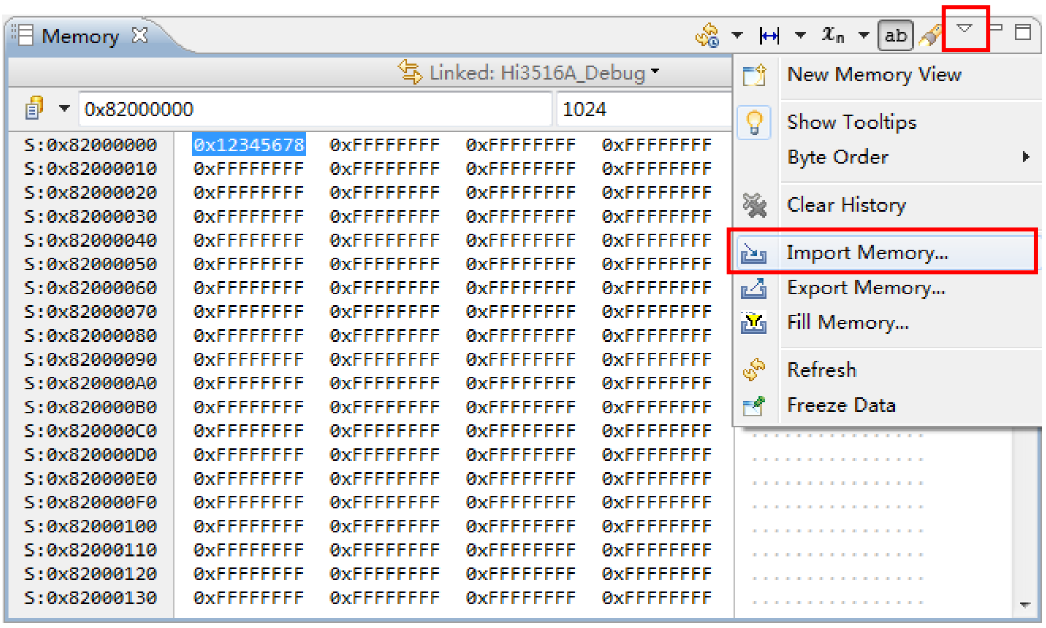
**CAUTION**

The .log file in **osdrv\tools\pc\uboot_tools** is the memory initialization script.

### 4.4.2 Downloading the U-boot Image

Perform the following steps:

**Step 1** Click in the **Memory** window, see Figure 4-13.



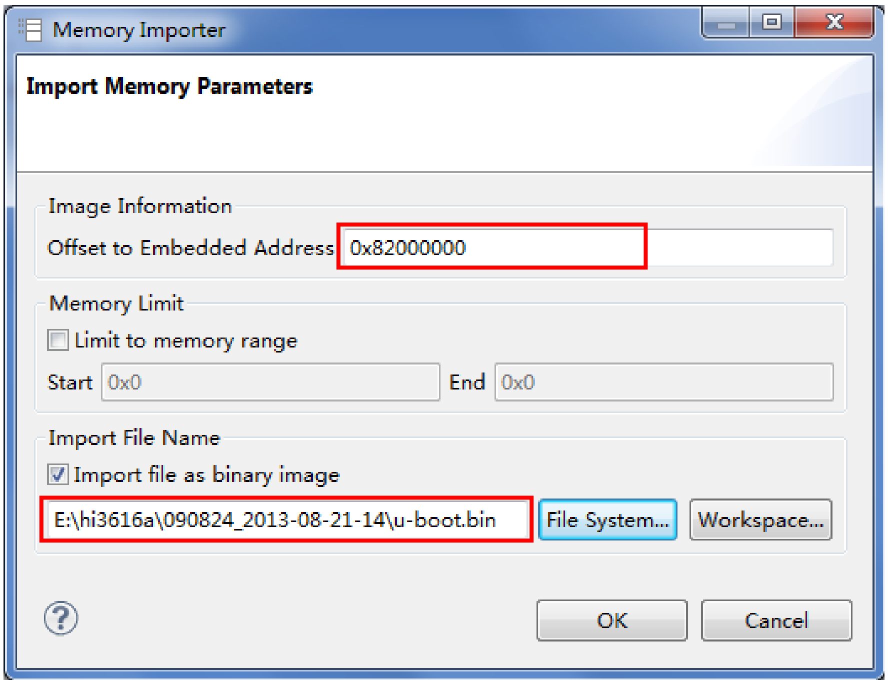
```
0x82000000                                        1024
S:0x82000000   0x12345678   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000010   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000020   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000030   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000040   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000050   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000060   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000070   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000080   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000090   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x820000A0   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x820000B0   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x820000C0   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x820000D0   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x820000E0   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x820000F0   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000100   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000110   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000120   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
S:0x82000130   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF   0xFFFFFFFF
```

**Figure 4-13** Memory window

**Step 2** Select **Import Memory**. The image download window is displayed.

**Step 3** Download the U-boot image to the memory address such as **0x82000000**. See Figure 4-14.

**Figure 4-14** Memory Import window



Memory Importer

Import Memory Parameters

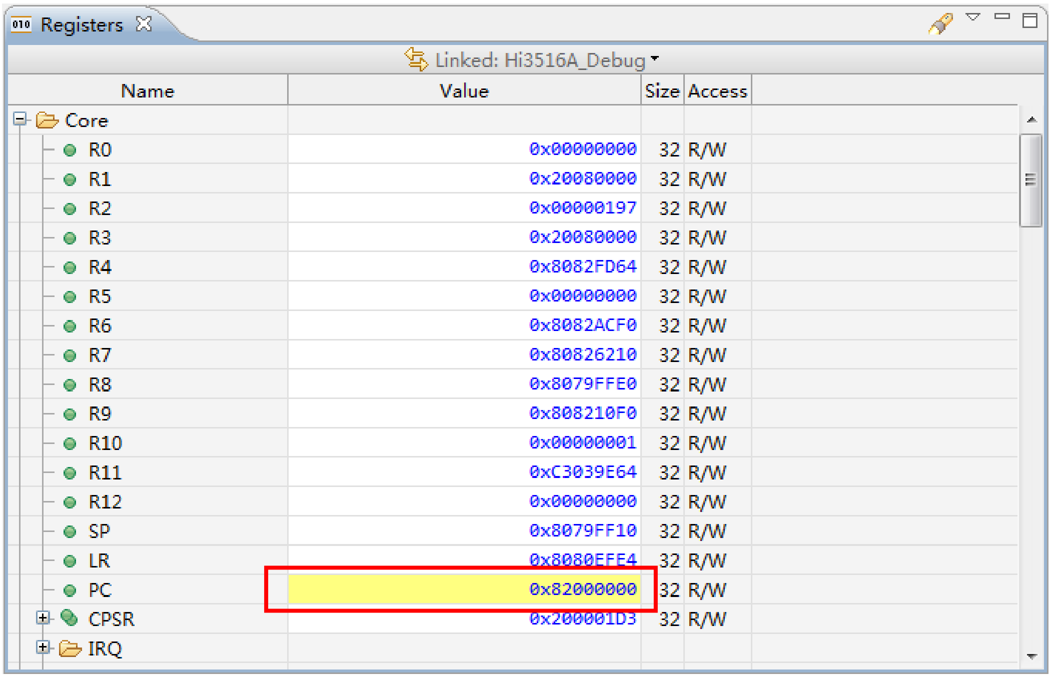
Image Information
Offset to Embedded Address: 0x82000000

Memory Limit
☐ Limit to memory range
Start: 0x0 End: 0x0

Import File Name
☑ Import file as binary image
E:\hi3616a\090824_2013-08-21-14\u-boot.bin

**Step 4** In the **Registers** window, change the PC value to **0x82000000**. See Figure 4-15.

**Figure 4-15** Registers window



| Name | Value | Size | Access |
|---|---|---|---|
| Core | | | |
| R0 | 0x00000000 | 32 | R/W |
| R1 | 0x20080000 | 32 | R/W |
| R2 | 0x00000197 | 32 | R/W |
| R3 | 0x20080000 | 32 | R/W |
| R4 | 0x8082FD64 | 32 | R/W |
| R5 | 0x00000000 | 32 | R/W |
| R6 | 0x8082ACF0 | 32 | R/W |
| R7 | 0x80826210 | 32 | R/W |
| R8 | 0x8079FFE0 | 32 | R/W |
| R9 | 0x808210F0 | 32 | R/W |
| R10 | 0x00000001 | 32 | R/W |
| R11 | 0xC3039E64 | 32 | R/W |
| R12 | 0x00000000 | 32 | R/W |
| SP | 0x8079FF10 | 32 | R/W |
| LR | 0x8080EFE4 | 32 | R/W |
| PC | 0x82000000 | 32 | R/W |
| CPSR | 0x200001D3 | 32 | R/W |
| IRQ | | | |

**Step 5** Click ▶ in the **Debug Control** window to start the U-boot, and view the U-boot start information over the serial port.

**----End**

### 4.4.3 Burning the U-boot Image

After the U-boot starts, burn the U-boot image in the memory to a flash memory over the serial port.

Taking SPI flash as an example, the burning commands are as follows:

```
hisilicon# sf probe 0

hisilicon# sf erase 0 100000

hisilicon# sf write 82000000 0 100000

hisilicon# reset
```

# 5 Appendix

## 5.1 U-boot Commands

### 5.1.1 Block Protection Command for the SPI Flash

The SPI NOR flash provides the block protection (BP) bits to ensure data security.

A corresponding block in the component enters the write protection status by setting the BP0, BP1, BP2, or BP3 (the status register of some chips may have BP4 but not BP3) bit in the status register (SR) to **1** (enabled). These BP bits are non-volatile and can retain the configured status during power-off.

All the BP bits in the SRs for SPI components are 0 (disabled) by default. The BP for all the blocks in the components is disabled and the blocks can be erased or written.

When all the BP bits are set to 1 (enabled), all the blocks in the components are in write protection status and the erase and write operations have no effect.

The number of consecutive blocks (starting from block 0 or block 511) whose BP is enabled depends on the BP level. The BP level ranges from BP[0:0:0:0] to BP[1:1:1:1]. When the BP level is 1 (BP[0:0:0:1]), the BP of one block is enabled. When the BP level is 2 (BP[0:0:1:0]), the BP of two blocks is enabled. When the BP level is 3 (BP[0:0:1:1]), the BP of four blocks is enabled. When the BP level is 4 (BP[0:1:0:0]), the BP of eight blocks is enabled. The number of blocks whose BP is enabled doubles when the BP level increases by 1 until the BP of all the blocks in the component is enabled, as shown in Table 5-1.

**Table 5-1** Mapping between the size of protected blocks and the BP level when the BP of blocks at the bottom of the component is enabled first

| **Size of protected blocks (T/B bit = 1)** | | | | |
|---|---|---|---|---|
| **Block Protection Bit** | | | | **Protection Level** |
| **BP3** | **BP2** | **BP1** | **BP0** | **256 Mbits** |
| 0 | 0 | 0 | 0 | 0 (none) |
| 0 | 0 | 0 | 1 | 1 (1 blocks, block 0 protected) |
| 0 | 0 | 1 | 0 | 2 (2 blocks, block 0–block 1 protected) |
| 0 | 0 | 1 | 1 | 3 (4 blocks, block 0–block 3 protected) |

| Size of protected blocks (T/B bit = 1) | | | | |
|---|---|---|---|---|
| Block Protection Bit | | | | Protection Level |
| BP3 | BP2 | BP1 | BP0 | 256 Mbits |
| 0 | 1 | 0 | 0 | 4 (8 blocks, block 0–block 7 protected) |
| 0 | 1 | 0 | 1 | 5 (16 blocks, block 0–block 15 protected) |
| 0 | 1 | 1 | 0 | 6 (32 blocks, block 0–block 31 protected) |
| 0 | 1 | 1 | 1 | 7 (64 blocks, block 0–block 63 protected) |
| 1 | 0 | 0 | 0 | 8 (128 blocks, block 0–block 127 protected) |
| 1 | 0 | 0 | 1 | 9 (256 blocks, block 0–block 255 protected) |
| 1 | 0 | 1 | 0 | 10 (512 blocks, all blocks protected) |
| 1 | 0 | 1 | 1 | 11 (512 blocks, all blocks protected) |
| 1 | 1 | 0 | 0 | 12 (512 blocks, all blocks protected) |
| 1 | 1 | 0 | 1 | 13 (512 blocks, all blocks protected) |
| 1 | 1 | 1 | 0 | 14 (512 blocks, all blocks protected) |
| 1 | 1 | 1 | 1 | 15 (512 blocks, all blocks protected) |

Some vendors provide the function of configuring the BP direction, that is, whether to enable the BP of blocks from the top of the component or from the bottom of the component can be configured. The BP direction configuration bit is one time programming (OTP) type and its default value is 0. When the BP of blocks at the top of the component (upper address) is enabled first, the BP direction configuration bit is set to **1**. Then if the BP of blocks at the bottom of the component (lower address) is enabled first, the bit value cannot be changed. Table 5-2 shows the mapping between the size of protected blocks and the BP level when the BP of blocks at the top of the component is enabled first.

**Table 5-2** Mapping between the size of protected blocks and the BP level when the BP of blocks at the top of the component is enabled first

| Size of protected blocks (T/B bit = 0) | | | | |
|---|---|---|---|---|
| Block Protection Bit | | | | Protection Level |
| BP3 | BP2 | BP1 | BP0 | 256 Mbits |
| 0 | 0 | 0 | 0 | 0 (none) |
| 0 | 0 | 0 | 1 | 1 (1 blocks, block 511 protected) |
| 0 | 0 | 1 | 0 | 2 (2 blocks, block 510–block 511 protected) |
| 0 | 0 | 1 | 1 | 3 (4 blocks, block 508–block 511 protected) |
| 0 | 1 | 0 | 0 | 4 (8 blocks, block 504–block 511 protected) |
| 0 | 1 | 0 | 1 | 5 (16 blocks, block 496–block 511 protected) |

| Size of protected blocks (T/B bit = 0) | | | | |
|---|---|---|---|---|
| Block Protection Bit | | | | Protection Level |
| BP3 | BP2 | BP1 | BP0 | 256 Mbits |
| 0 | 1 | 1 | 0 | 6 (32 blocks, block 480–block 511 protected) |
| 0 | 1 | 1 | 1 | 7 (64 blocks, block 448–block 511 protected) |
| 1 | 0 | 0 | 0 | 8 (128 blocks, block 384–block 511 protected) |
| 1 | 0 | 0 | 1 | 9 (256 blocks, block 256–block 511 protected) |
| 1 | 0 | 1 | 0 | 10 (512 blocks, all blocks protected) |
| 1 | 0 | 1 | 1 | 11 (512 blocks, all blocks protected) |
| 1 | 1 | 0 | 0 | 12 (512 blocks, all blocks protected) |
| 1 | 1 | 0 | 1 | 13 (512 blocks, all blocks protected) |
| 1 | 1 | 1 | 0 | 14 (512 blocks, all blocks protected) |
| 1 | 1 | 1 | 1 | 15 (512 blocks, all blocks protected) |

The **lock** BP command is added for SPI components in U-boot based on the BP mechanism for the SPI NOR flash. The command format can be **sf lock all**, **sf lock [t/b]** <**level**>, or **sf lock**.

The parameters in the **lock** commands are described as follows:

- **all**: All the blocks are protected. In this case, the BP level is 10, as shown in Table 5-1.
- **[t/b]**: The BP direction is configured. When [t/b] is b, the BP of blocks at the bottom of the component is enabled first, as shown in Table 5-1. When [t/b] is t, the BP of blocks at the top of the component is enabled first, as shown in Table 5-2.

  Note that [t/b] is optional and the default value is b, indicating that the BP of blocks at the bottom of the component is enabled first. An OTP bit is configured for [t/b]. Therefore, when the BP direction changes from top to bottom, a warning message is displayed, asking you to confirm the change.
- <**level**>: The BP level is configured. Corresponding blocks are protected according to the BP level. The effect of different BP levels is described as follows:
  - When the BP level is 0, the BP of all the blocks is disabled. In this case, configuring **[t/b]** has no effect.
  - When the BP level is 10, the BP of all the blocks is enabled. The effect is equivalent to that of parameter **all**. In this case, configuring **[t/b]** has no effect.
  - When the BP level ranges from 1 to 9, certain blocks are protected and the BP direction depends on [t/b]. The number of protected blocks doubles when the BP level increases by 1. The number of protected blocks ranges from 1 to half the size of the component. The BP direction needs to be changed to protect the other half blocks.

The configured BP level, size of protected blocks, and BP direction are displayed, and the command description can be displayed by running **sf lock**.