Hi3536D V100 U-Boot Porting

Development Guide

Issue **00B02**

Date **2017-11-20**

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 how to port and burn the U-Boot (that is, bootloader of the Hi3536D V100 board) on the Hi3536D V100 board, and how to use ARM debugging tools.

Related Version

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

Product Name	Version
Hi3536D	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 00B02(2017-11-20)

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

Chapter 2 Porting the U-Boot

Section 2.5is updated.

Issue 00B01 (2017-09-08)

This issue is the first draft release.

Contents

1 Overview......1

1.1 Description......1

1.2 U-Boot Directory Structure......1

2 Porting the U-Boot......2

2.1 U-Boot Hardware Environment......2

2.2 Compiling the U-Boot......2

2.3 Configuring the DDR......3

2.4 Configuring Pin Multiplexing......3

2.5 Generating the Final U-Boot Image......3

3 Burning the U-Boot......4

3.1 Overview......4

3.2 Burning the U-Boot by Using the BOOTROM......4

3.3 Burning the U-Boot to Two Types of Flash Memories......4

3.3.1 Burning the U-Boot to the SPI NOR Flash......4

3.3.2 Burning the U-Boot to the SPI NAND Flash......5

4 Using ARM Debugging Tools......6

4.1 Overview......6

4.2 ARM Debugging Tools......6

4.2.1 DS-5 Eclipse......6

4.2.2 DS-5 Debug......6

4.3 Using ARM Debugging Tools......7

4.3.1 Installing the DS-5......7

4.3.2 Creating a Configuration Database for the Target Platform......8

4.3.3 Connecting the DS-5 to the Target Platform......14

4.4 Burning Images to a Flash Memory by Using the Simulator......17

4.4.1 Initializing the Memory......17

4.4.2 Downloading the U-Boot Image......19

4.4.3 Burning the U-Boot Image......21

5 Appendix......22

5.1 U-Boot Commands......22

5.1.1 Block Protection Command for the SPI NOR Flash......22

Figures

Figure 4-1 Startup GUI of the DS-5 Eclipse 8

Figure 4-2 Platform configuration window 9

Figure 4-3 Config IP window 9

Figure 4-4 Config IP scan window 10

Figure 4-5 Window for setting the IP address of the simulator 10

Figure 4-6 Window for creating the platform database 11

Figure 4-7 Window for creating the platform database 11

Figure 4-8 Window for creating the platform database 12

Figure 4-9 Window for creating the platform database 12

Figure 4-10 Window for creating the platform database 13

Figure 4-11 Window for creating the platform database 13

Figure 4-12 Window for creating the platform database 14

Figure 4-13 End Debug Configuration window 15

Figure 4-14 Debug Configuration window 15

Figure 4-15 Debug Configuration window 16

Figure 4-16 DS-5 Debug – Eclipse Platform window 17

Figure 4-17 Scripts window 18

Figure 4-18 Memory window 19

Figure 4-19 Memory window 19

Figure 4-20 Memory Import window 20

Figure 4-21 Registers window 21

Figure 5-1 Viewing current BP information 24

Figure 5-2 Locking the entire component 25

Figure 5-3 Clearing the current locking status 25

Figure 5-4 Locking the specified area by setting the level 25

Tables

Table 1-1 Main directory structure of the U-Boot 1

Table 5-1 Mapping between the BP levels and BP lock areas of components of different vendors 23

1 Overview

1.1 Description

The bootloader of the Hi3536D V100 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.

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/Hi3536dv100	Indicates the code of the Hi3536D V100 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.
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 Hi3536D V100 demo board include the DDR SDRAM, SPI NOR flash, and SPI NAND flash. For details, see the *Hi3536D V100 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- hi3536dv100_config
```

NOTE

hi3536dv100_config is used for the configuration of the SPI NOR flash and SPI NAND flash.

Step 2 Compile the U-Boot.

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

If the compilation is successful, the **u-boot.bin** file will be generated in the **u-boot** directory.

NOTE

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

- Hi3536D_V100R001C01SPCxxx corresponds to uclibc. If the uclibc tool chain is used, the **CROSS_COMPILE** parameter is set to **arm-hisiv510-linux-**.
- Hi3536D_V100R001C02SPCxxx corresponds to glibc. If the glibc tool chain is used, the **CROSS_COMPILE** parameter is set to **arm-hisiv610-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_dmc0**, **mddrc_dmc1**, **mddrc_phy0,** and **mddrc_phy1** 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 the configuration sheet is modified.

Step 2 Click **Generate reg bin file** on the first tab page of the configuration sheet or use the hiregbin tool (for details, see the readme file in **osdrv/ tools/pc/uboot_tools/ hiregbin-v5.0.0.tgz**) to generate the temporary file **reg_info.bin**.

Step 3 Copy **u-boot.bin** (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.bin u-boot-hi3536dv100.bin
```

u-boot-hi3536dv100.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 Hi3536D V100 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 *HiTool 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 NOR 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          /*Initialize the memory.*/
hisilicon# tftp 0x82000000 u-boot-hi3536dv100.bin /*Download the U-Boot to the memory.*/
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 memory to the SPI NOR
flash.*/
```

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

----End

CAUTION

In the current issue, **sf lock** can be used to protect blocks of the SPI NOR Flash. If a block of the SPI NOR Flash is protected, the block becomes read-only, the **erase** and **write** commands do not take effect, and the block is still protected even after power outrage. In this case, the erase and write commands are valid only after the **sf lock 0** command is executed. For details, see section 5.1.1 "Block Protection Command for the SPI NOR Flash."

3.3.2 Burning the U-Boot to the SPI 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 0x0 0x100000    /*Erase 1 MB capacity of the NAND flash.*/
hisilicon# mw.b  0x82000000  ff  0x100000        /*Initialize  the  memory.*/
hisilicon# tftp  0x82000000 u-boot-hi3536dv100.bin     /*Download the u-boot
to  the  memory.*/
hisilicon# nand write 0x82000000 0x0 0x100000   /*Write the U-Boot from the
memory to the SPI NAND flash.*/
```

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

----End

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.

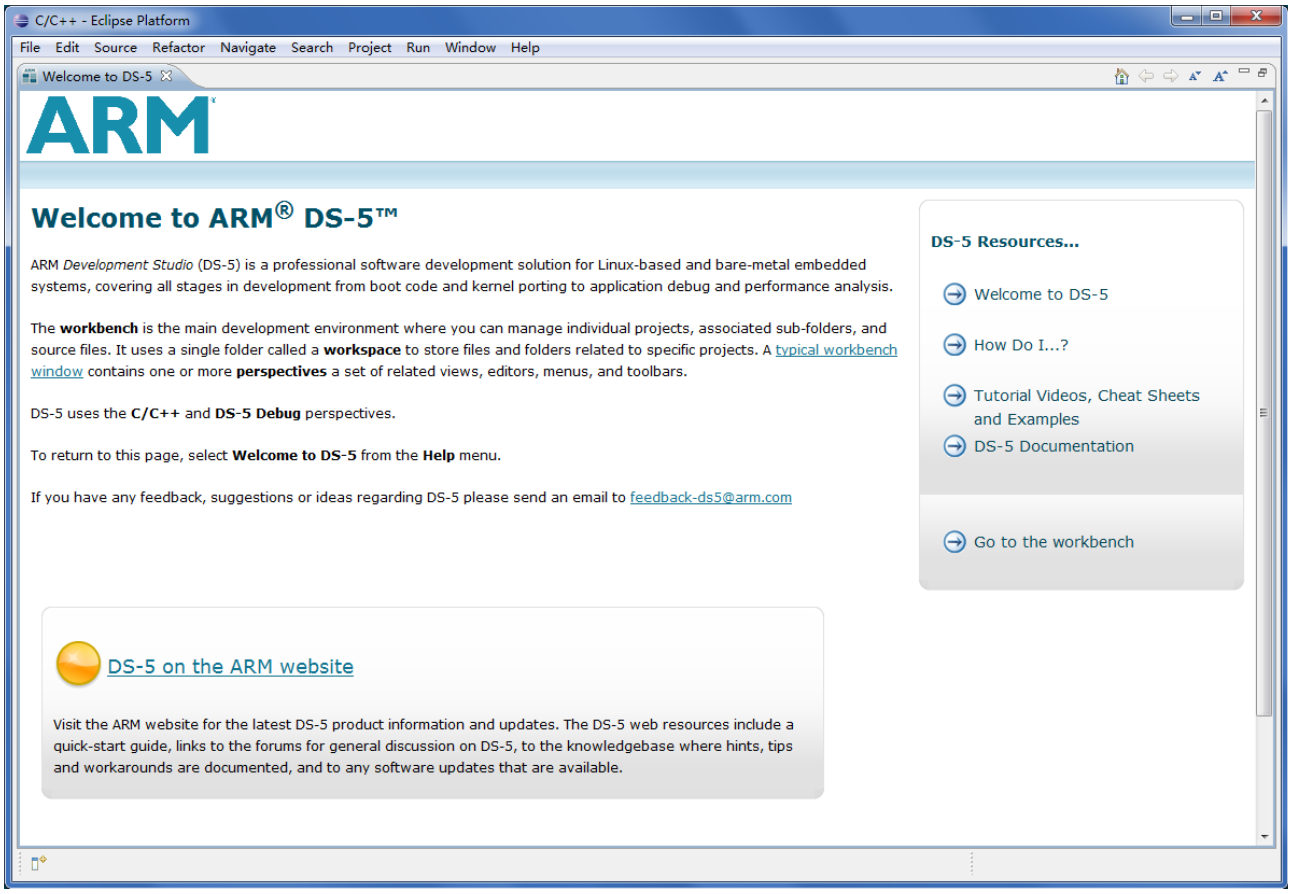
Step 3 Create a new connection to connect the DS-5 to the target platform by using the configuration database for the target platform.

----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 Resources...

Welcome to DS-5

How Do I...?

Tutorial Videos, Cheat Sheets and Examples

DS-5 Documentation

Go to the workbench

DS-5 on the ARM website

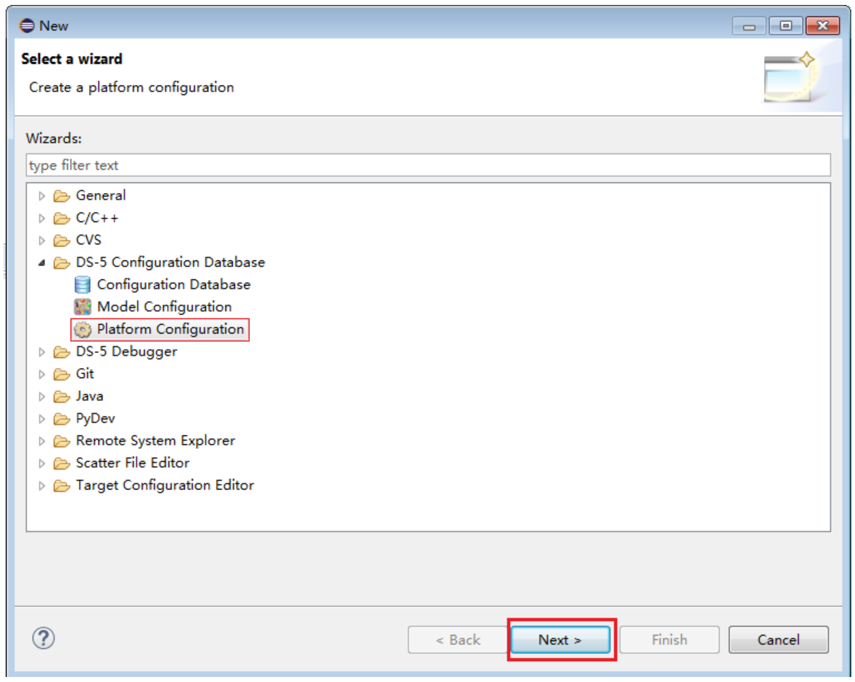
Visit the ARM website for the latest DS-5 product information and updates. The DS-5 web resources include a quick-start guide, links to the forums for general discussion on DS-5, to the knowledgebase where hints, tips and workarounds are documented, and to any software updates that are available.

4.3.2 Creating a Configuration Database for the Target Platform

Perform the following steps:

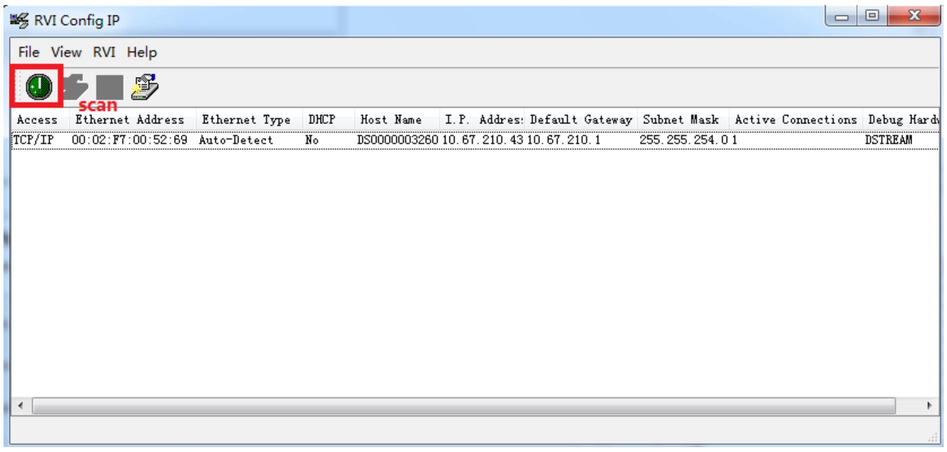
Step 1 Choose **File** > **New** > **Other**. In the displayed dialog box, select **Platform Configuration** under **DS-5 Configuration Database**, and then click **Next >**, as shown in Figure 4-2.

Figure 4-2 Platform configuration window



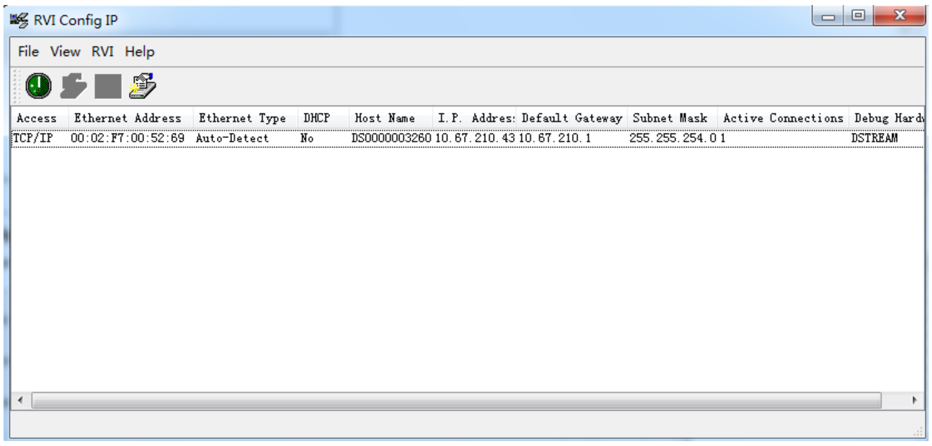
Step 2 Connect to the simulator. Choose **ARM DS-5 v5.24.1** > **Debug Hardware** > **Debug Hardware Config IP(5.24.1)** from the menu. In the displayed software window, click **scan**. After the simulator is found, set an IP address that is in the same network segment as the current PC, as shown in Figure 4-3 and Figure 4-5.

Figure 4-3 Config IP window



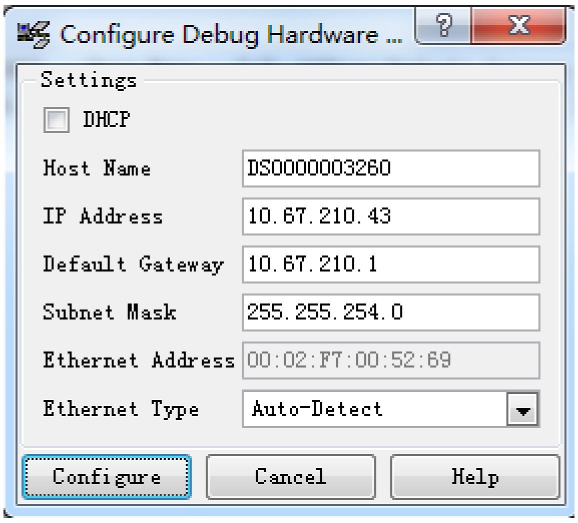
Access	Ethernet Address	Ethernet Type	DHCP	Host Name	I.P. Addres:	Default Gateway	Subnet Mask	Active Connections	Debug Hardw
TCP/IP	00:02:F7:00:52:69	Auto-Detect	No	DS0000003260	10.67.210.43	10.67.210.1	255.255.254.0	1	DSTREAM

Figure 4-4 Config IP scan window



Access	Ethernet Address	Ethernet Type	DHCP	Host Name	I.P. Addres:	Default Gateway	Subnet Mask	Active Connections	Debug Hardw
TCP/IP	00:02:F7:00:52:69	Auto-Detect	No	DS0000003260	10.67.210.43	10.67.210.1	255.255.254.0	1	DSTREAM

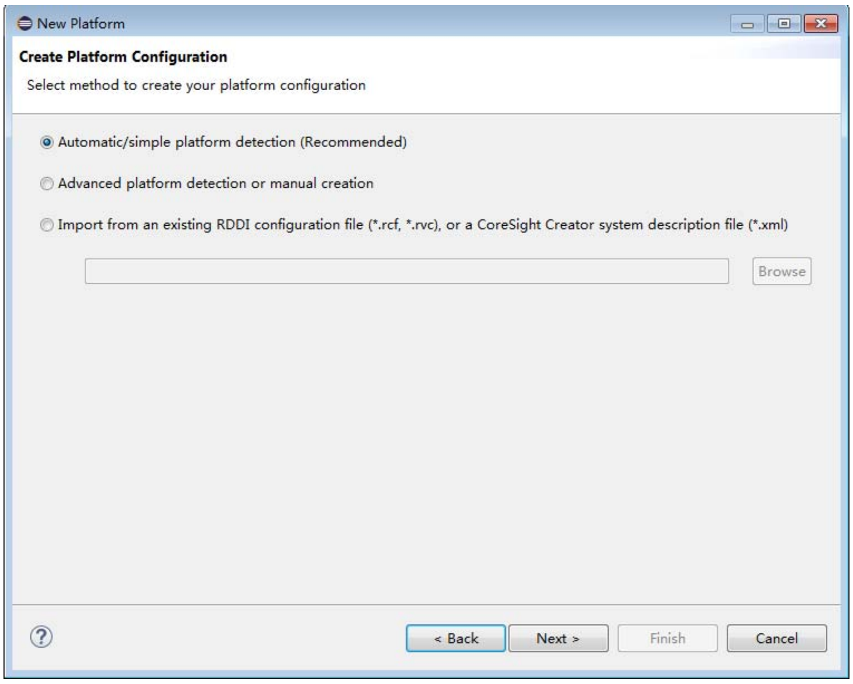
Figure 4-5 Window for setting the IP address of the simulator



DHCP	☐
Host Name	DS0000003260
IP Address	10.67.210.43
Default Gateway	10.67.210.1
Subnet Mask	255.255.254.0
Ethernet Address	00:02:F7:00:52:69
Ethernet Type	Auto-Detect

Step 3 Go back to the **DS-5 Eclipse** window. Select **Automatic/simple platform detection(Recommended)**, and click **Next >**. The system performs the scan. Enter the IP address of the simulator in **Connection Address:** and click **Next >**. Select **Debug target after saving configuration**, and click **Next >**. Click **Create New Database**. Enter the database name and click **OK**. Then, click **Next >**. Set **Platform Manufacturer** to **Hisilicon**, set **Platform Name** to **Hi3536D V100**, and click **Finish**, as shown in Figure 4-6 to Figure 4-12.

Figure 4-6 Window for creating the platform database



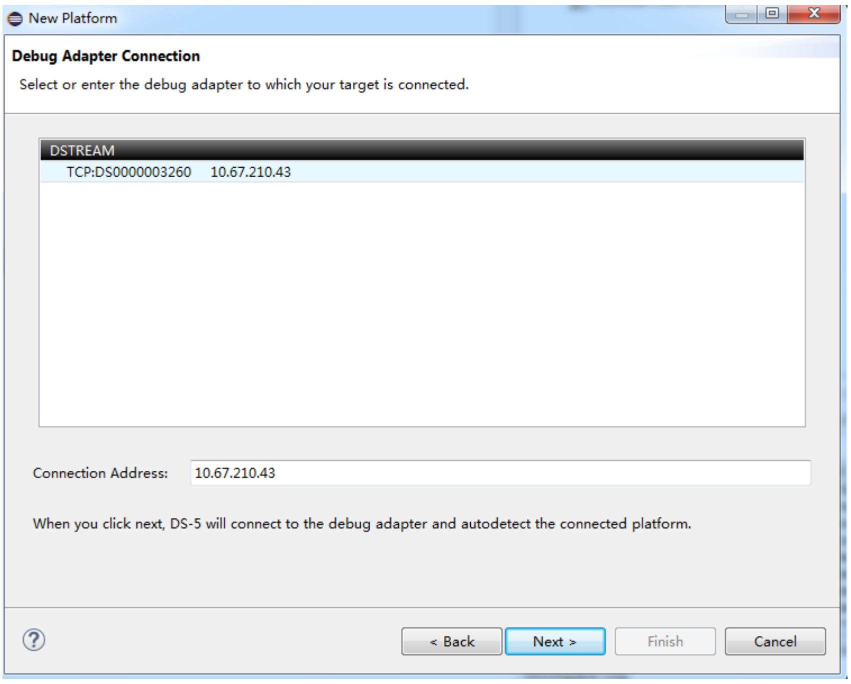
Select method to create your platform configuration

(●) Automatic/simple platform detection (Recommended)

( ) Advanced platform detection or manual creation

( ) Import from an existing RDDI configuration file (*.rcf, *.rvc), or a CoreSight Creator system description file (*.xml)

Figure 4-7 Window for creating the platform database



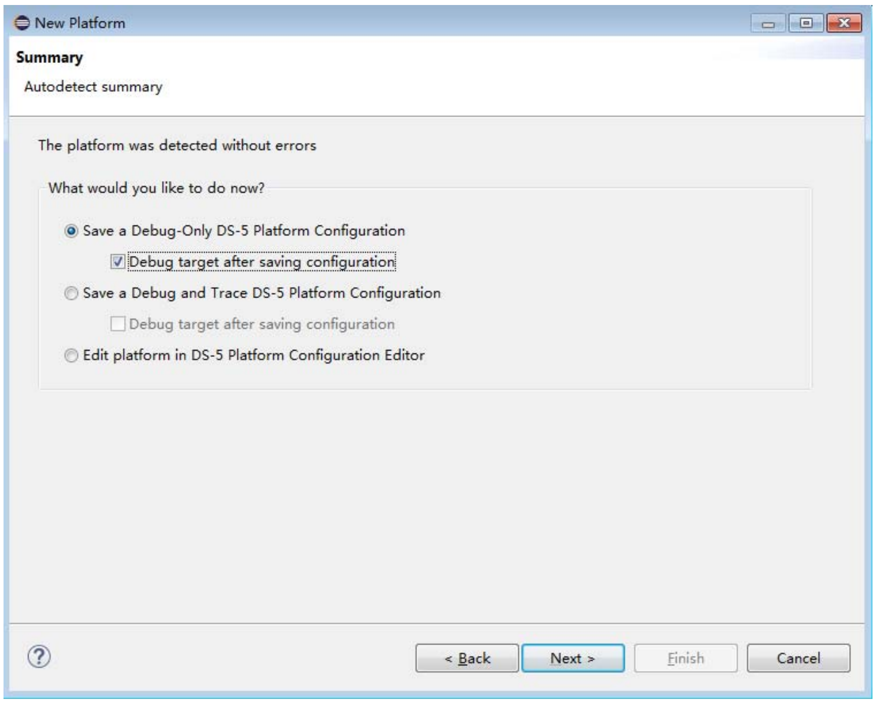
Select or enter the debug adapter to which your target is connected.

DSTREAM
TCP:DS0000003260 10.67.210.43

Connection Address: 10.67.210.43

When you click next, DS-5 will connect to the debug adapter and autodetect the connected platform.

Figure 4-8 Window for creating the platform database



Summary

Autodetect summary

The platform was detected without errors

What would you like to do now?

(●) Save a Debug-Only DS-5 Platform Configuration

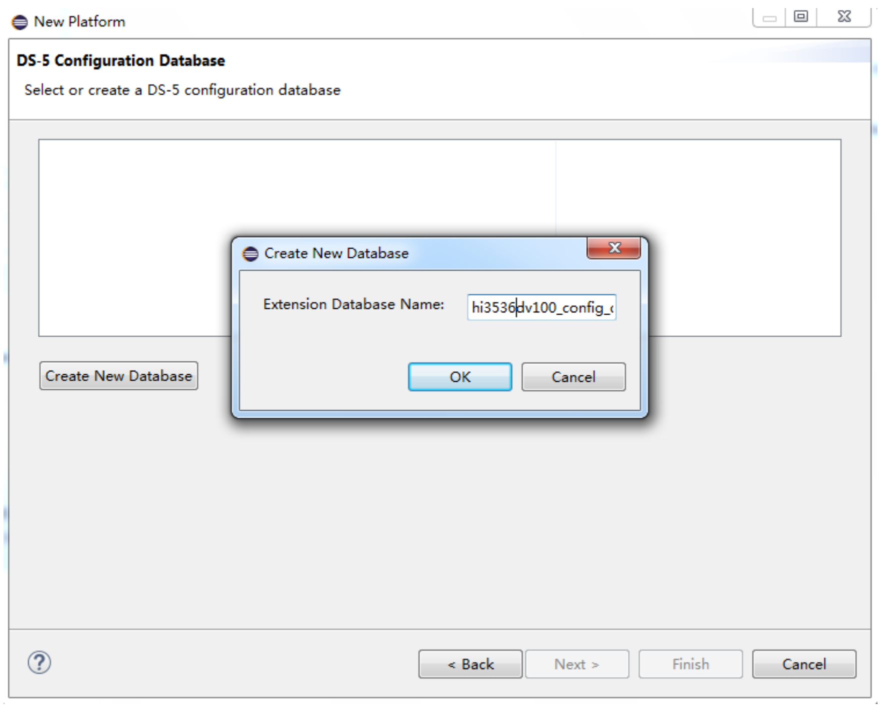
[✓] Debug target after saving configuration

( ) Save a Debug and Trace DS-5 Platform Configuration

[ ] Debug target after saving configuration

( ) Edit platform in DS-5 Platform Configuration Editor

Figure 4-9 Window for creating the platform database



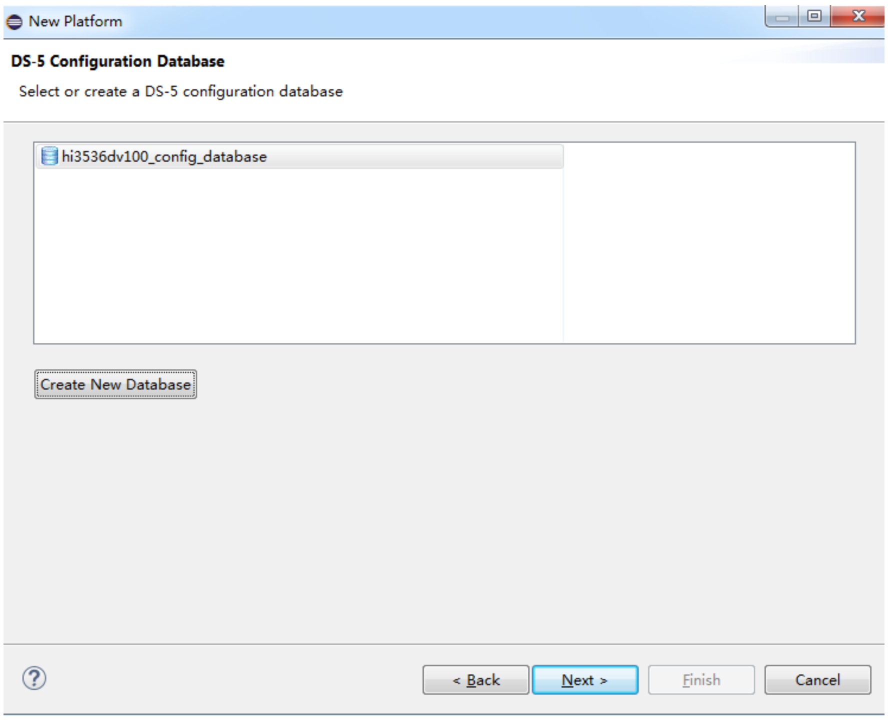
DS-5 Configuration Database

Select or create a DS-5 configuration database

Create New Database

Extension Database Name: hi3536dv100_config_

Figure 4-10 Window for creating the platform database

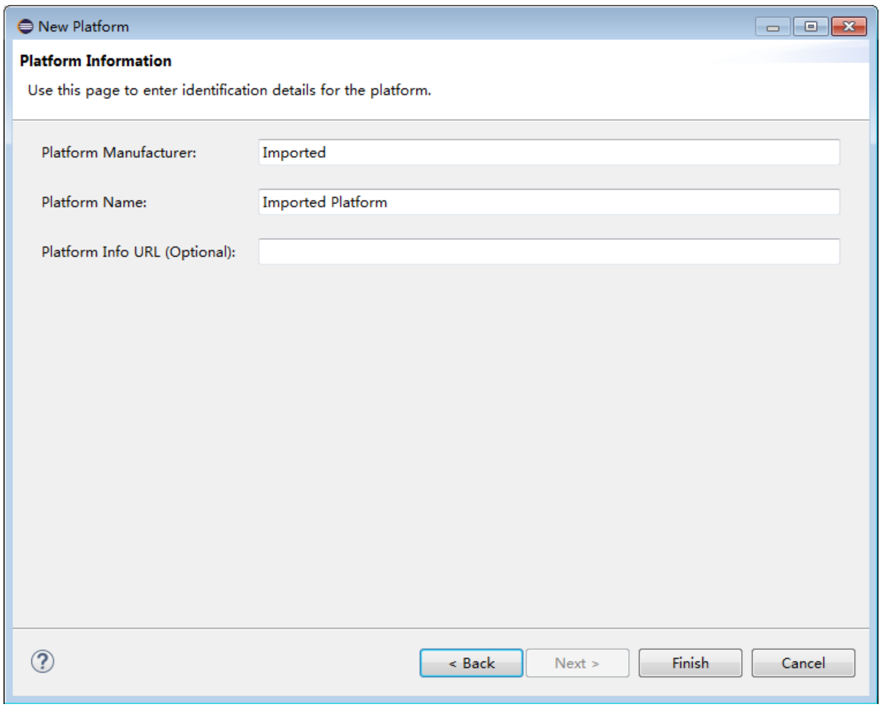

New Platform

DS-5 Configuration Database

Select or create a DS-5 configuration database

hi3536dv100_config_database

Figure 4-11 Window for creating the platform database


New Platform

Platform Information

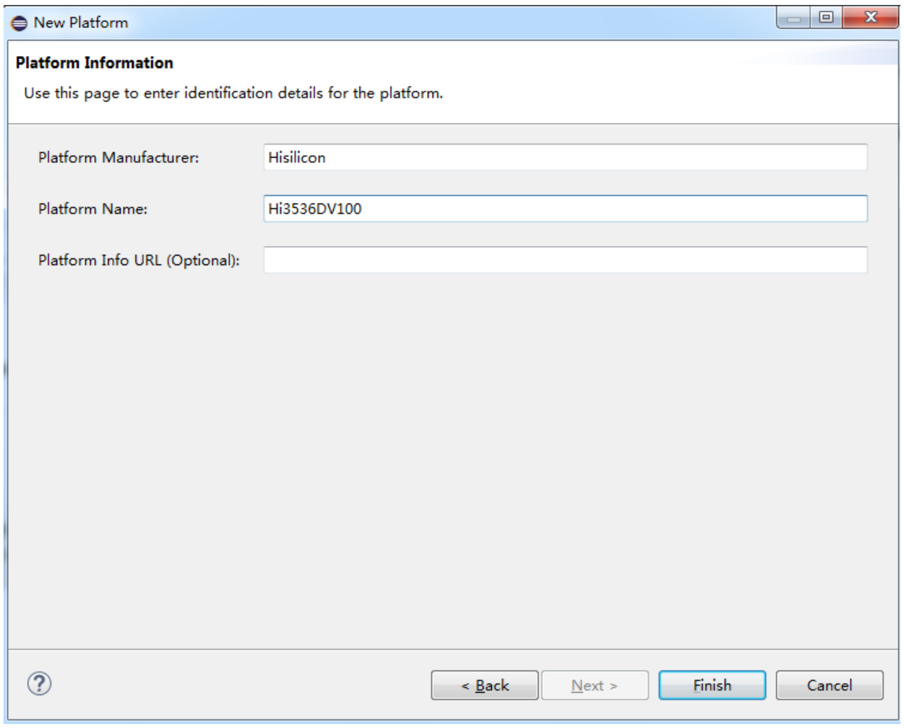
Use this page to enter identification details for the platform.

Platform Manufacturer: Imported

Platform Name: Imported Platform

Platform Info URL (Optional):

Figure 4-12 Window for creating the platform database



Platform Information

Use this page to enter identification details for the platform.

Platform Manufacturer: Hisilicon

Platform Name: Hi3536DV100

Platform Info URL (Optional):

----End

4.3.3 Connecting the DS-5 to the Target Platform

Perform the following steps:

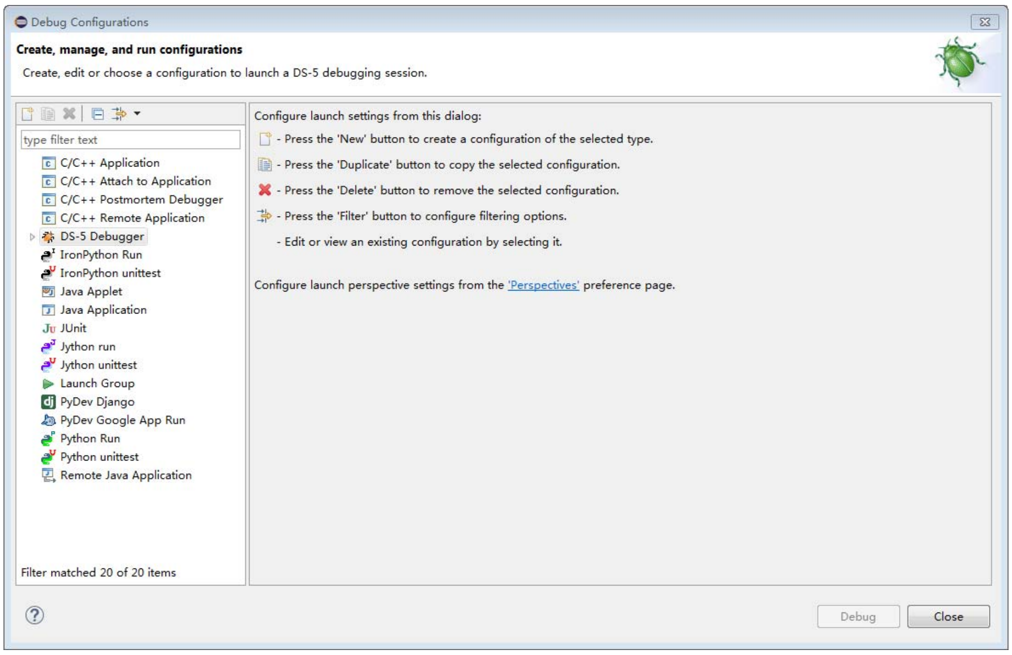
Step 1 In the displayed dialog box after **Finish** is clicked, right-click **DS-5 Debugger**, choose **New** from the shortcut menu, and select the created **Hisilicon-Hi3536D V100**, as shown in Figure 4-13.

Step 2 On the **Connection** tab page, click **Hisilicon**, **Hi3536D V100**, **Bare Metal Debug**, and **Cortex-A7**, and enter the IP address of the DS-5 device in the text box, as shown in Figure 4-14.

Step 3 Select **Connect Only** on the **Debugger** tab page, as shown in Figure 4-15.

Step 4 Click **Debug** to connect to the target platform. The window shown in Figure 4-16 is displayed.

Figure 4-13 End Debug Configuration window



Create, manage, and run configurations

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

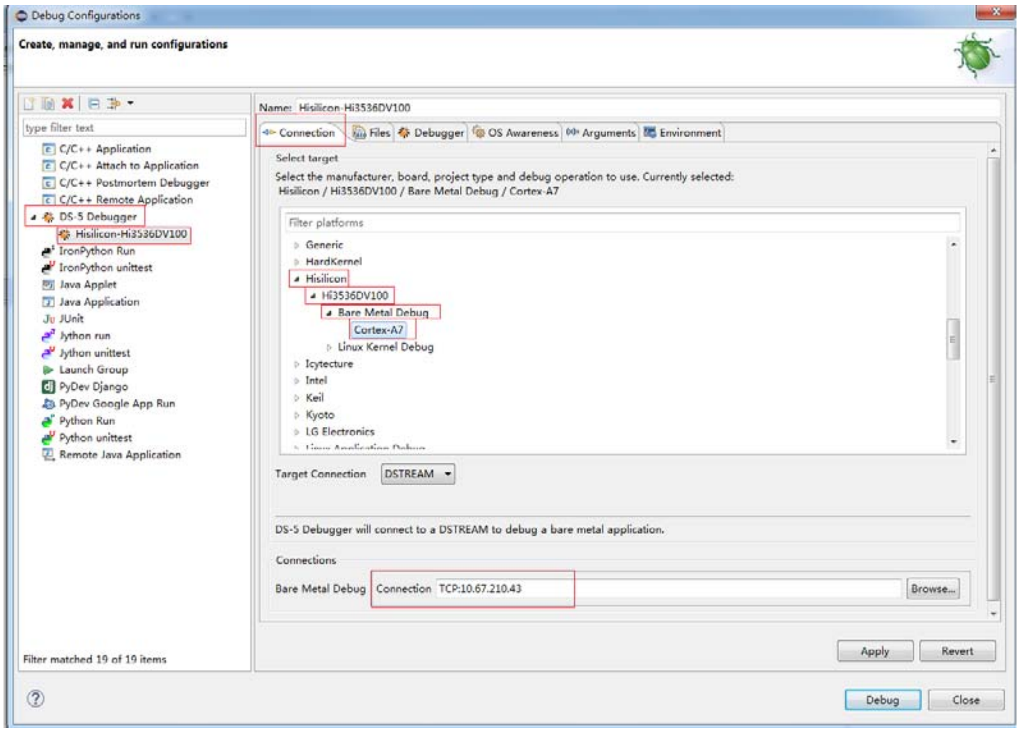
Configure launch settings from this dialog:

- 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.

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

Filter matched 20 of 20 items

Figure 4-14 Debug Configuration window



Create, manage, and run configurations

Name: Hisilicon-Hi3536DV100

Select target

Select the manufacturer, board, project type and debug operation to use. Currently selected:
Hisilicon / Hi3536DV100 / Bare Metal Debug / Cortex-A7

Target Connection: DSTREAM

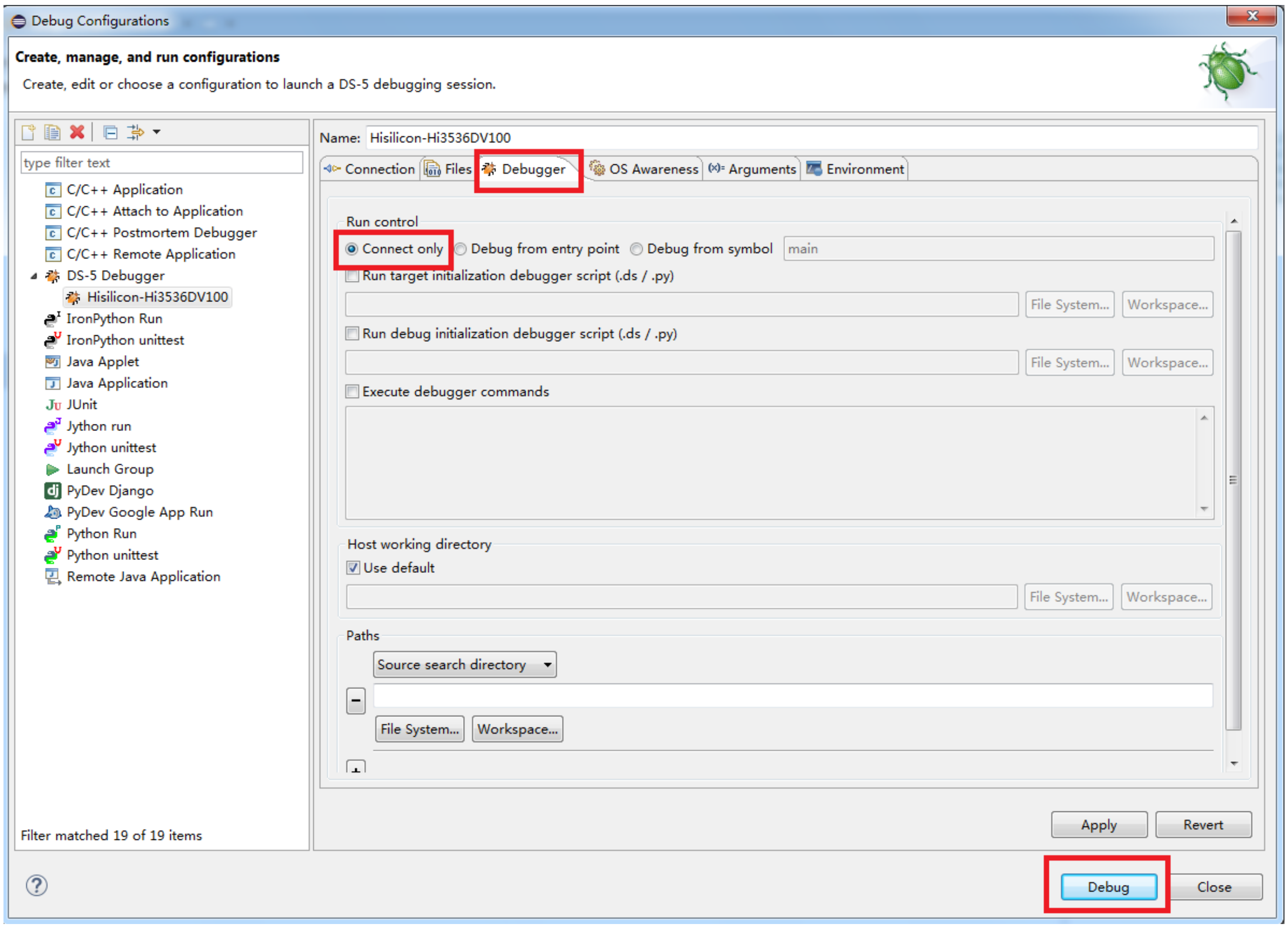
DS-5 Debugger will connect to a DSTREAM to debug a bare metal application.

Connections

Bare Metal Debug Connection TCP:10.67.210.43

Filter matched 19 of 19 items

Figure 4-15 Debug Configuration 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
  - Hisilicon-Hi3536DV100
- IronPython Run
- IronPython 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: Hisilicon-Hi3536DV100

Run control

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

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

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

☐ Execute debugger commands

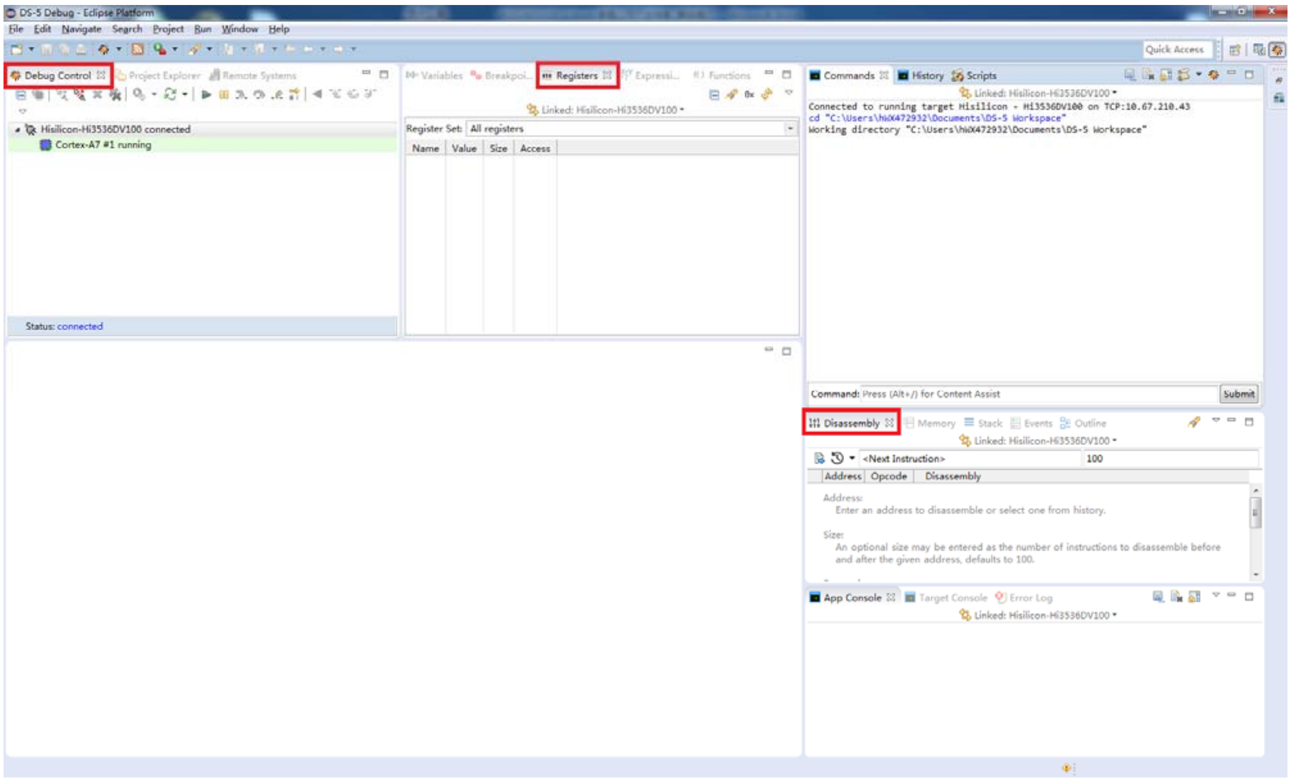
Host working directory

☑ Use default

Paths

Source search directory

Figure 4-16 DS-5 Debug – Eclipse Platform window



```
Connected to running target Hisilicon - Hi3536DV100 on TCP:10.67.210.43
cd "C:\Users\hWX472932\Documents\DS-5 Workspace"
Working directory "C:\Users\hWX472932\Documents\DS-5 Workspace"
```

Address:
Enter an address to disassemble or select one from history.

Size:
An optional size may be entered as the number of instructions to disassemble before and after the given address, defaults to 100.

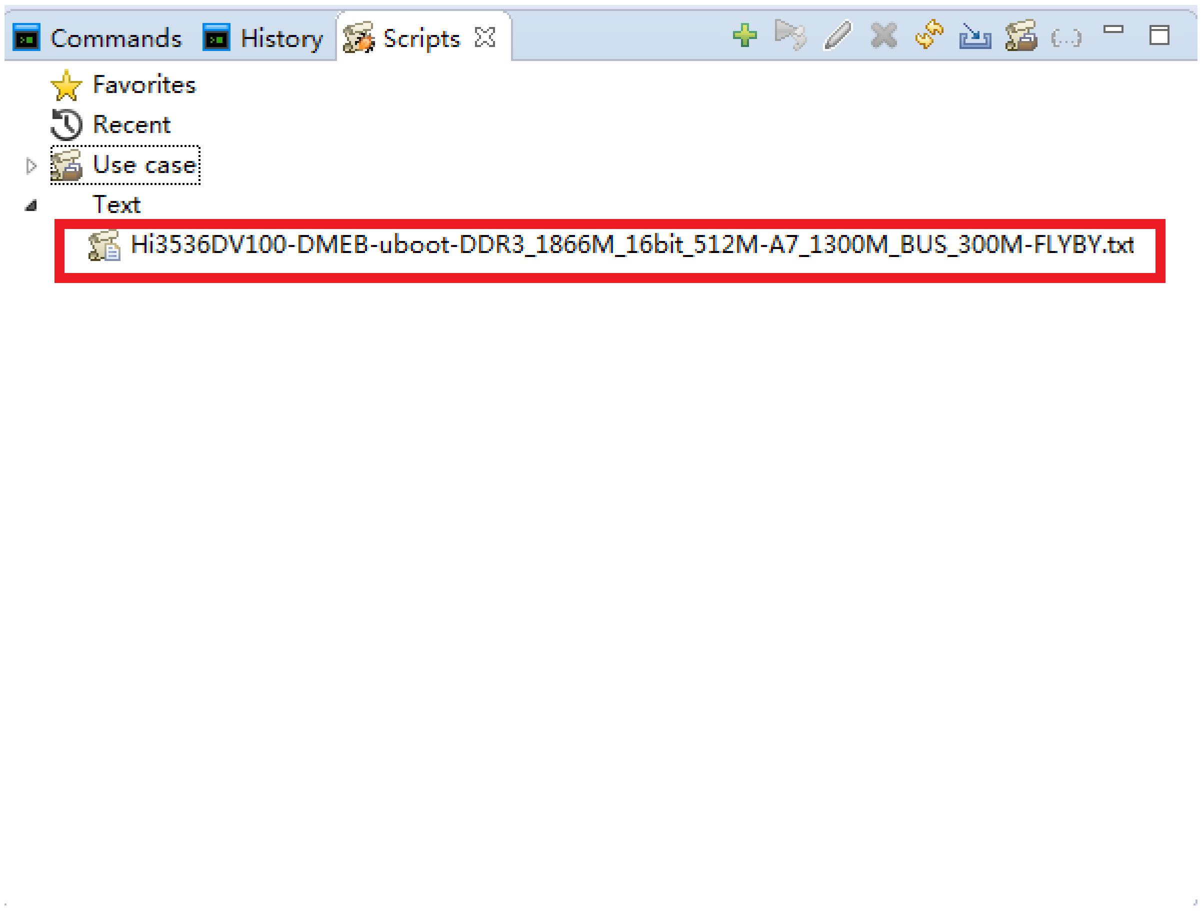
----End

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 4-17.

Figure 4-17 Scripts window

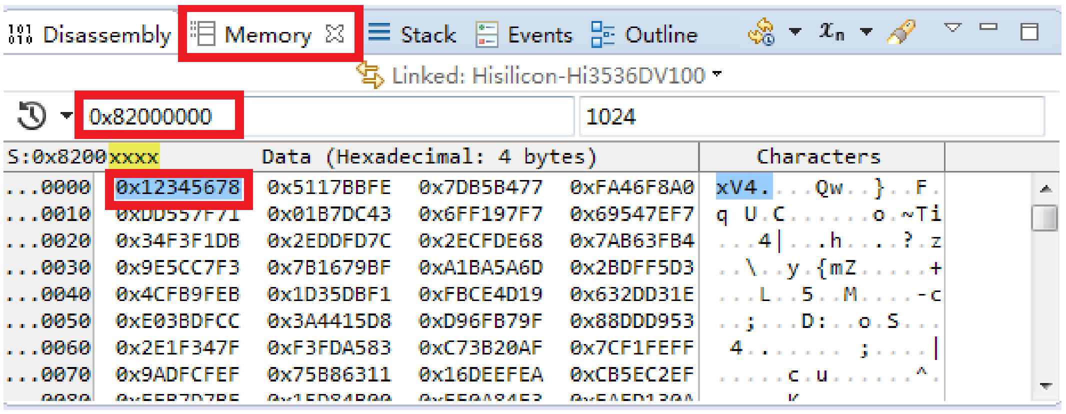


Hi3536DV100-DMEB-uboot-DDR3_1866M_16bit_512M-A7_1300M_BUS_300M-FLYBY.txt

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 4-18. 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 4-18.

Figure 4-18 Memory window



```
0x82000000                                   1024
S:0x8200xxxx      Data (Hexadecimal: 4 bytes)          Characters
...0000  0x12345678  0x5117BBFE  0x7DB5B477  0xFA46F8A0  xV4....Qw..}..F.
...0010  0xDD557F71  0x01B7DC43  0x6FF197F7  0x69547EF7  q U.C.......o.~Ti
...0020  0x34F3F1DB  0x2EDDFD7C  0x2ECFDE68  0x7AB63FB4  ...4|...h....?.z
...0030  0x9E5CC7F3  0x7B1679BF  0xA1BA5A6D  0x2BDFF5D3  ..\..y.{mZ......+
...0040  0x4CFB9FEB  0x1D35DBF1  0xFBCE4D19  0x632DD31E  ...L..5..M.....-c
...0050  0xE03BDFCC  0x3A4415D8  0xD96FB79F  0x88DDD953  ..;...D:..o.S...
...0060  0x2E1F347F  0xF3FDA583  0xC73B20AF  0x7CF1FEFF   4.......;....|
...0070  0x9ADFCFEF  0x75B86311  0x16DEEFEA  0xCB5EC2EF  ......c.u......^.
```

CAUTION

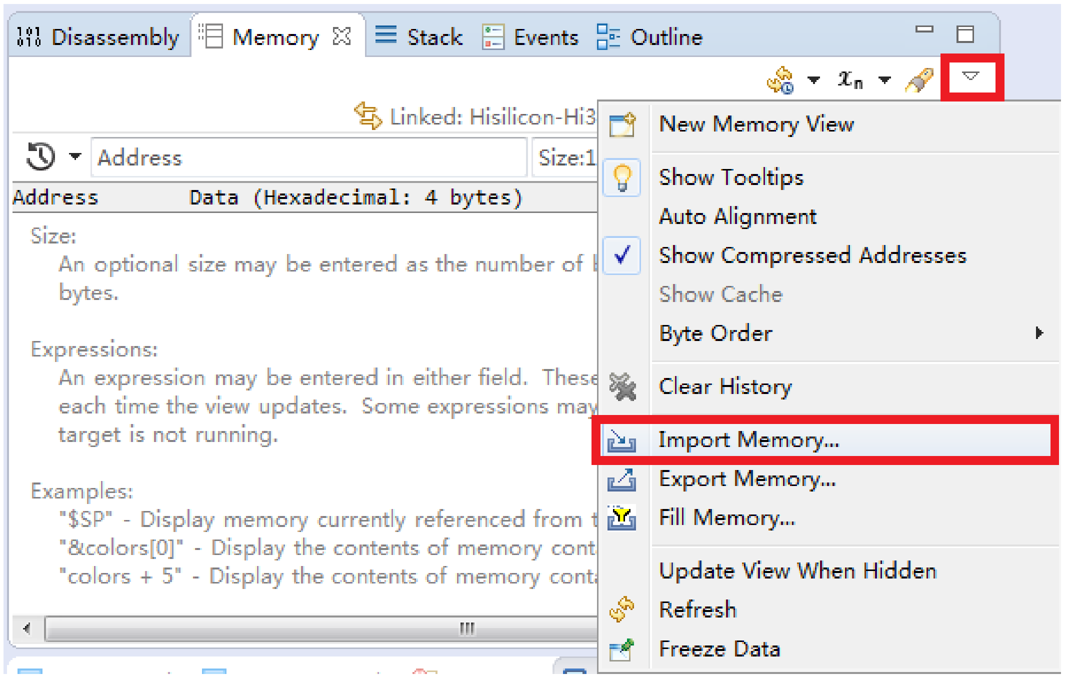
The script for memory initialization is the file in .ds, **.py** or **.txt** format in the **osdrv\tools\pc\uboot_tools** directory.

4.4.2 Downloading the U-Boot Image

Perform the following steps:

Step 1 Click ▽ in the **Memory** window, see Figure 4-19.

Figure 4-19 Memory window



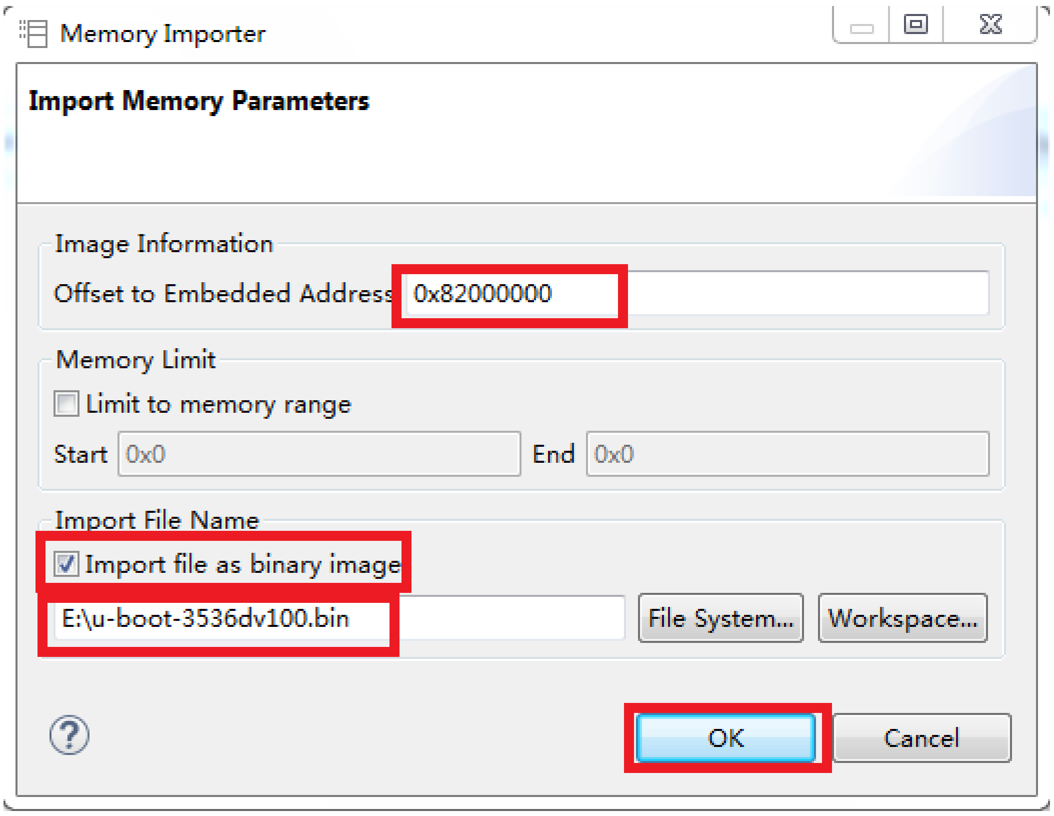
Size:
An optional size may be entered as the number of b
bytes.

Expressions:
An expression may be entered in either field. These
each time the view updates. Some expressions may
target is not running.

Examples:
"$SP" - Display memory currently referenced from t
"&colors[0]" - Display the contents of memory cont
"colors + 5" - Display the contents of memory cont

Step 2 Select **Import Memory**. The image download window is displayed. Download the U-Boot image to the memory address such as **0x82000000**. See Figure 4-20.

Figure 4-20 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

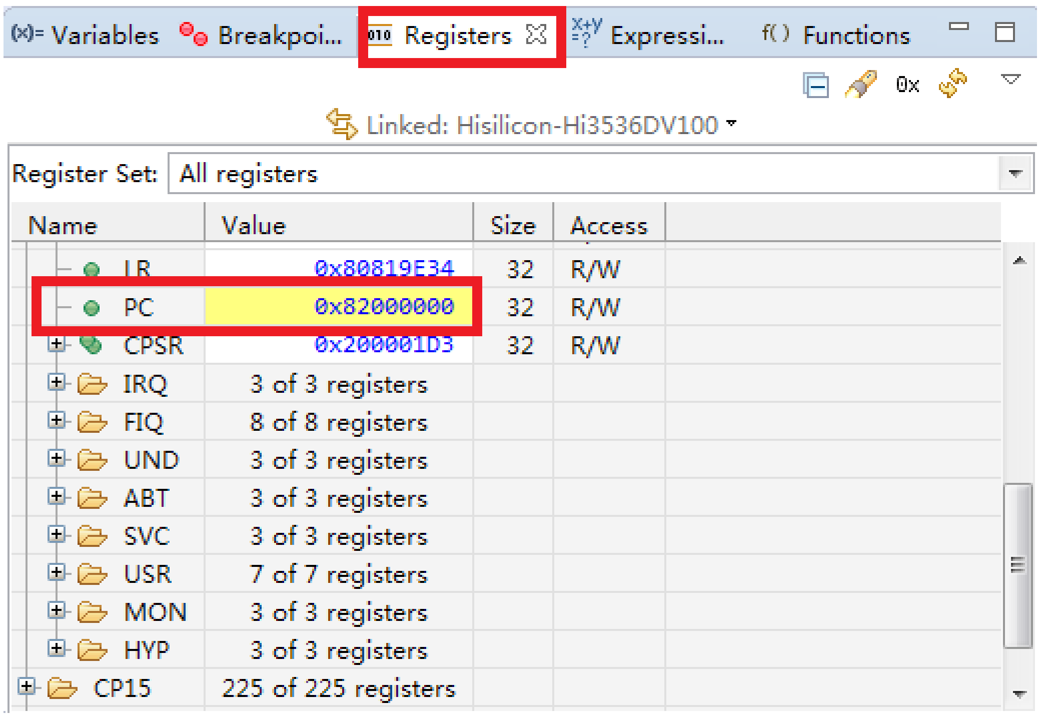
Import File Name

☑ Import file as binary image

E:\u-boot-3536dv100.bin

Step 3 In the **Registers** window, change the PC value to **0x82000000**. See Figure 4-21.

Figure 4-21 Registers window



Register Set: All registers

Name	Value	Size	Access
LR	0x80819E34	32	R/W
PC	0x82000000	32	R/W
CPSR	0x200001D3	32	R/W
IRQ	3 of 3 registers		
FIQ	8 of 8 registers		
UND	3 of 3 registers		
ABT	3 of 3 registers		
SVC	3 of 3 registers		
USR	7 of 7 registers		
MON	3 of 3 registers		
HYP	3 of 3 registers		
CP15	225 of 225 registers		

Step 4 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 NOR flash as an example, the burning commands are as follows:

```
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
memory to the SPI flash. */
hisilicon# reset                                /* Restart the board. */
```

5 Appendix

5.1 U-Boot Commands

5.1.1 Block Protection Command for the SPI NOR 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.

Some vendors provide the function of configuring the BP direction, that is, whether to enable the BP function of blocks from the top of the component or from the bottom of the component can be configured. Whether the start address for write protection locking starts from the low address (bottom) or high address (top) is configured in the TBPROT bit (for the chips of some vendors, the TBPROT bit is located in the SR) of the configuration register (CR). Typically this configuration bit is the one-time programmable (OTP) type. The default bit value is 0, indicating that the BP starts from the top (high address). Once this bit is set to 1, indicating that the BP starts from the bottom (low address), the status cannot be changed any more.

Based on actual applications, the TBPROT is set to 1 since initialization of the controller, that is, BP starts from the bottom (low address).

All the BP bits in the SRs for SPI NOR 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 BP is implemented based on the unit of block. The hexadecimal value of each BP bit is converted into the decimal level value based on the enable status of the BP bits to configure the range of locked BP. For the components with three BP bits, the value range of level is 0–7 for BP[0:0:0] to BP[1:1:1]. For the components with four BP bits, the value range of level is 0–10 (or 0–9 because the minimum locked area cannot be less than one block) for BP[0:0:0:0] to BP[1:1:1:1].

The BP lock area varies according to the components of different vendors, as shown in Table 5-1.

Table 5-1 Mapping between the BP levels and BP lock areas of components of different vendors

Level	**MXIC MX25L-**					**ESMT F25L-**
	12835F	**25635F**	**25735F**	**6406E**	**1606E**	**64QA**
0	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked
1	0–64 KB	0–64 KB	0–64 KB	0–4 MB (64 blocks)	All-locked 2 MB (32 blocks)	0–4 MB (64 blocks)
2	0–128 KB	0–128 KB	0–128 KB	0–6 MB (96 blocks)	0–1 MB (16 blocks)	0–6 MB (96 blocks)
3	0–256 KB	0–256 KB	0–256 KB	0–7 MB (112 blocks)	0–1.5 MB (24 blocks)	0–7 MB (112 blocks)
4	0–512 KB	0–512 KB	0–512 KB	0–7.5 MB (120 blocks)	0–1.75 MB (28 blocks)	0–7.5 MB (120 blocks)
5	0–1 MB	0–1 MB	0–1 MB	0–7.75 MB (124 blocks)	0–1.875 MB (30 blocks)	0–7.75 MB (124 blocks)
6	0–2 MB	0–2 MB	0–2 MB	0–7.875 MB (126 blocks)	0–1.9375 MB (31 blocks)	0–7.875 MB (126 blocks)
7	0–4 MB	0–4 MB	0–4 MB	All-locked 8 MB (128 blocks)	All-locked 2 MB (32 blocks)	All-locked 8 MB (128 blocks)
8	0–8 MB	0–8 MB	0–8 MB	-	-	-
9	All-locked 16 MB	0–16 MB	0–16 MB	-	-	-
10	-	All-locked 32 MB	All-locked 32 MB	-	-	-

Level	**SPANSION S25FL-**		**WINBOND W25Q-**			
	127S	**256S**	**128FV**	**128BV**	**256FV**	**64FV**
0	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked
1	0–256 KB	0–512 KB	0–256 KB	0–256 KB	0–64 KB	0–128 KB
2	0–512 KB	0–1 MB	0–512 KB	0–512 KB	0–128 KB	0–256 KB
3	0–1 MB	0–2 MB	0–1 MB	0–1 MB	0–256 KB	0–512 KB
4	0–2 MB	0–4 MB	0–2 MB	0–2 MB	0–512 KB	0–1 MB
5	0–4 MB	0–8 MB	0–4 MB	0–4 MB	0–1 MB	0–2 MB
6	0–8 MB	0–16 MB	0–8 MB	0–8 MB	0–2 MB	0–4 MB
7	All-locked 16 MB	All-locked 32 MB	All-locked 16 MB	All-locked 16 MB	0–4 MB	All-locked 8 MB

8	-	-	-	-	0–8 MB	-
9	-	-	-	-	0–16 MB	-
10	-	-	-	-	All-locked 32 MB	-

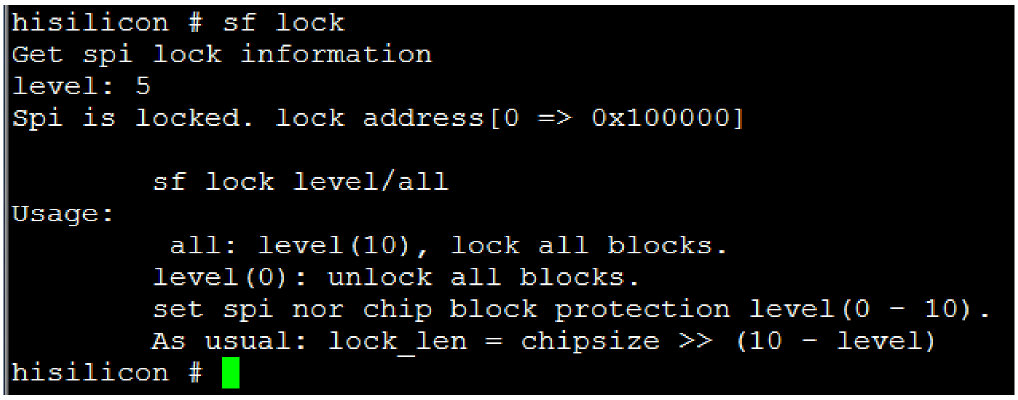
Level	**GD GD25Q-**			**CFEON EN25Q-**	
	128C	**64**	**32**	**128**	**64**
0	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked
1	0–256 KB	0–128 KB	0–64 MB	0–15.9375 MB (255 blocks)	0–7.9375 MB (127 blocks)
2	0–512 KB	0–256 KB	0–128 MB	0–15.875 MB (254 blocks)	0–7.875 MB (126 blocks)
3	0–1 MB	0–512 KB	0–256 MB	0–15.75 MB (252 blocks)	0–7.75 MB (124 blocks)
4	0–2 MB	0–1 MB	0–512 MB	15.5 MB(248 blocks)	0–7.5 MB (120 blocks)
5	0–4 MB	0–2 MB	0–1 MB	0–15 MB(240 blocks)	0–7 MB (112 blocks)
6	0–8 MB	0–4 MB	0–2 MB	0–14 MB(224 blocks)	0–6 MB (96 blocks)
7	All-locked 16 MB	All-locked 8 MB	All-locked 4 MB	All-locked 16 MB (256 blocks)	All-locked 8 MB (128 blocks)

The **lock** BP command is added for SPI NOR components in U-Boot based on the BP mechanism for the SPI NOR flash. The command formats are as follows:

- sf lock

  This command can be used to view the configured BP level, value range of the level, and the range of locked areas. At the same time, the command description information is displayed. See Figure 5-1.

Figure 5-1 Viewing current BP information



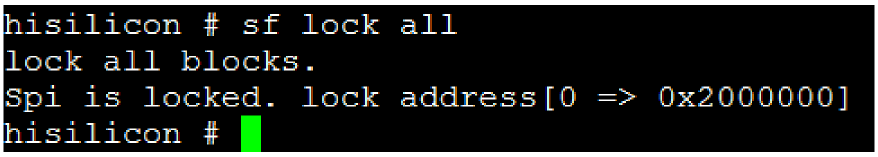
```
hisilicon # sf lock
Get spi lock information
level: 5
Spi is locked. lock address[0 => 0x100000]

        sf lock level/all
Usage:
         all: level(10), lock all blocks.
        level(0): unlock all blocks.
        set spi nor chip block protection level(0 - 10).
        As usual: lock_len = chipsize >> (10 - level)
hisilicon #
```

- sf lock all

This command is used to lock all the blocks (of the entire component). As shown in Table 5-1, running this command is equivalent to setting the level to the maximum value. See Figure 5-2.

Figure 5-2 Locking the entire component

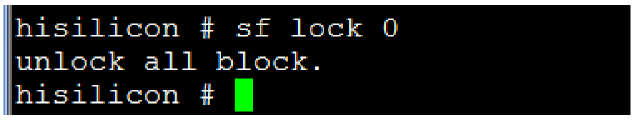


```
hisilicon # sf lock all
lock all blocks.
Spi is locked. lock address[0 => 0x2000000]
hisilicon #
```

- sf lock 0

  This command is used to clear the current BP locking status. In this case, all the blocks in the component are not protected, and can be erased. See Figure 5-3.

Figure 5-3 Clearing the current locking status

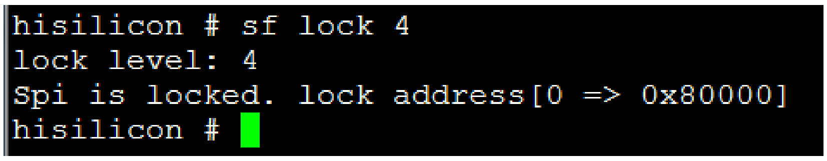


```
hisilicon # sf lock 0
unlock all block.
hisilicon #
```

- sf lock <level>

  This command is used to set the BP level. The corresponding areas are protected based on the level, as described in Table 5-1. In this way, the blocks in the BP area cannot be normally erased. See Figure 5-4.

Figure 5-4 Locking the specified area by setting the level



```
hisilicon # sf lock 4
lock level: 4
Spi is locked. lock address[0 => 0x80000]
hisilicon #
```