



# **Hi3516C V500/Hi3516D V300/Hi3516A V300 U-boot Porting Development Guide**


**Issue**            **00B03**

**Date**            **2018-11-20**

**Copyright © HiSilicon (Shanghai) Technologies Co., Ltd. 2019. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon (Shanghai) Technologies Co., Ltd.

## **Trademarks and Permissions**

 , **HISILICON** , and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **HiSilicon (Shanghai) Technologies Co., Ltd.**

Address: New R&D Center, 49 Wuhe Road, Bantian,  
Longgang District,  
Shenzhen 518129 P. R. China

Website: <http://www.hisilicon.com/en/>

Email: [support@hisilicon.com](mailto:support@hisilicon.com)



# About This Document

## Purpose

This document describes how to port and burn the U-Boot (that is, bootloader of the Hi3516C V500/Hi3516D V300 board) on the Hi3516C V500/Hi3516D V300/Hi3516A V300 board, and how to use ARM debugging tools.



### NOTE

This document uses Hi3516C V500 as an example. In the subsequent description, if Hi3516D V300/Hi3516A V300 is used, replace Hi3516C V500 with Hi3516D V300/Hi3516A V300, and replace hi3516cv500 with hi3516dv300/hi3516av300.

## Related Versions

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

Product Name	Version
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300

## Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

## Change History

Changes between document issues are cumulative. The latest document issue contains all changes made in previous issues.



### **Issue 00B03 (2018-11-20)**

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

Section 2.5 is modified.

### **Issue 00B02 (2018-10-15)**

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

Sections 2.1, 3.3.2, and 4.4.1 are modified.

### **Issue 00B01 (2018-08-17)**

This issue is the first draft release.



# Contents

<b>About This Document.....</b>	<b>ii</b>
<b>1 Overview.....</b>	<b>1</b>
1.1 Description .....	1
1.2 U-Boot Directory Structure .....	1
<b>2 Porting the U-Boot .....</b>	<b>3</b>
2.1 U-Boot Hardware Environment .....	3
2.2 Compiling the U-Boot.....	3
2.3 Configuring the DDR SDRAM.....	4
2.4 Configuring Pin Multiplexing .....	4
2.5 Generating the Final U-Boot Image .....	4
<b>3 Burning the U-Boot.....</b>	<b>5</b>
3.1 Overview .....	5
3.2 Burning the U-Boot by Using the BOOTROM.....	5
3.3 Burning the U-Boot to Flash Memories .....	5
3.3.1 Burning the U-Boot to the SPI NOR Flash .....	5
3.3.2 Burning the U-Boot to the SPI NAND Flash .....	6
3.4 Burning the U-Boot to the eMMC .....	6
<b>4 Using ARM Debugging Tools.....</b>	<b>8</b>
4.1 Overview .....	8
4.2 ARM Debugging Tools .....	8
4.2.1 DS-5 Eclipse .....	8
4.2.2 DS-5 Debug .....	9
4.3 Using ARM Debugging Tools .....	9
4.3.1 Installing the ARM DS-5 .....	9
4.3.2 Creating a Configuration Database for the Target Platform.....	10
4.3.3 Connecting the DS-5 to the Target Platform .....	18
4.4 Burning Images to the Flash by Using the Simulator.....	21
4.4.1 Initializing the Memory .....	21
4.4.2 Downloading the U-Boot Image .....	22
4.4.3 Burning the U-Boot Image.....	24
<b>5 Appendix .....</b>	<b>26</b>



---

5.1 U-Boot Command Description.....	26
5.1.1 BP Command for the SPI NOR Flash.....	26



## Figures

<b>Figure 4-1</b> Startup GUI of the DS-5 Eclipse .....	10
<b>Figure 4-2</b> Platform configuration.....	11
<b>Figure 4-3</b> Config IP interface.....	11
<b>Figure 4-4</b> Config IP interface.....	12
<b>Figure 4-5</b> Simulator IP address configuration.....	12
<b>Figure 4-6</b> Platform database configuration window 1—Create Platform Configuration.....	13
<b>Figure 4-7</b> Platform database configuration window 2—Debug Adapter Connection .....	13
<b>Figure 4-8</b> Platform database configuration window 3—Summary .....	14
<b>Figure 4-9</b> Platform database configuration window 4—DS-5 Configuration Database—Create New Database .....	15
<b>Figure 4-10</b> Platform database configuration window 5—DS-5 Configuration Database—Create New Database .....	16
<b>Figure 4-11</b> Platform database configuration window 6—Platform Information.....	17
<b>Figure 4-12</b> Platform database configuration window 7—Platform Information.....	18
<b>Figure 4-13</b> Debug Configurations window .....	19
<b>Figure 4-14</b> Debug Configurations window—selecting a new target platform configuration data base and entering the IP address of the DS-5 device.....	19
<b>Figure 4-15</b> Debug Configurations window—selecting Connect only.....	20
<b>Figure 4-16</b> DS-5 Debug – Eclipse Platform window .....	21
<b>Figure 4-17</b> Scripts window .....	22
<b>Figure 4-18</b> Memory window.....	22
<b>Figure 4-19</b> Memory drop-down list .....	23
<b>Figure 4-20</b> Memory Import window .....	23
<b>Figure 4-21</b> Registers window.....	24
<b>Figure 5-1</b> Checking the current BP information. ....	28
<b>Figure 5-2</b> Locking all blocks.....	29
<b>Figure 5-3</b> Unlocking all blocks .....	29



<b>Figure 5-4</b> Specifying a BP area by setting the lock level.....	29
---	----





---

## Tables

---

<b>Table 1-1</b> Main directory structure of the U-Boot .....	1
<b>Table 5-1</b> Mapping between the BP lock levels and BP lock areas of components from different vendors .....	27



# 1 Overview

## 1.1 Description

The bootloader of the Hi3516C V500 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	Provides the code of the chip architecture and the entry code of the U-Boot.
board	Provides the code of boards, such as the memory driver code.
board/hisilicon/hi3516cv500	Provides the code of the Hi3516C V500 board.
arch/xxx/lib	Provides the code of architecture, such as the common code of ARM and microprocessor without interlocked pipeline stages (MIPS) architecture.
include	Provides header files.
include/configs	Provides the configuration files of boards.
common	Provides the implementation files for functions or commands.
drivers	Provides the driver code of Ethernet ports, flash memories, and serial ports.
net	Provides the implementation files for network protocols.



Directory	Description
fs	Provides the implementation files for file systems.
product/hiupdate	Provides the implementation files for functions of SD card upgrade and USB upgrade
product/hiosd	Provides the implementation files for DEC interfaces and HDMI, and the VO and MIPI functions
product/hiaudio	Provides implementation files for the audio function



# 2 Porting the U-Boot

## 2.1 U-Boot Hardware Environment

Peripheral components on the Hi3516C V500 demo board include the DDR SDRAM, eMMC, SPI NOR flash, and SPI NAND flash. For details, see the *Hi3516C V500 Compatibility Test Report*.

## 2.2 Compiling the U-Boot

After the porting is finished, you can compile the U-Boot by performing the following steps:

**Step 1** Configure the compilation environment.

Run the following configuration command:

```
make ARCH=arm CROSS_COMPILE=arm-himixXXX-linux- hi3516cv500_config
```

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

```
make ARCH=arm CROSS_COMPILE=arm-himixXXX-linux- -j 20
```

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



### NOTE

When compiling U-Boot, you need to add two parameters after **make**, that is, **ARCH=arm** **CROSS\_COMPILE=arm-himixXXX-linux-**. **CROSS\_COMPILE** indicates the toolchain.

- When the uClibc toolchain is used, **CROSS\_COMPILE** indicates **arm-himix100-linux-**.
- When the glibc toolchain is used, **CROSS\_COMPILE** indicates **arm-himix200-linux-**.
- In this document, **arm-himixXXX-linux** is used to represent the two tool chains.

## NOTICE

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

----End



## 2.3 Configuring the DDR SDRAM

On Windows, open the configuration table in **osdrv/ tools/pc/uboot\_tools** of the SDK. If a different DDR SDRAM is used, you must modify the related sheets in the configuration table.

## 2.4 Configuring Pin Multiplexing

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

## 2.5 Generating the Final U-Boot Image

Perform the following steps:

- Step 1** Run the **make** command in the **tools/pc/hi\_gzip** directory of the OSDRV, and copy the **gzip** file in the generated **bin/** directory to the **arch/arm/cpu/armv7/hi3516cv500/hw\_compressed/** directory of U-Boot.
- Step 2** Save settings after modifying the configuration sheet.
- Step 3** Click **Generate reg bin file** in the first sheet of the table to generate the temporary file **reg\_info.bin**.
- Step 4** Copy **reg\_info.bin** to the source code directory of U-Boot and rename it as **.reg**. Copy the generated **u-boot.bin** to the **osdrv/tools/pc/uboot\_tools/** directory.
- Step 5** Run the **make ARCH=arm CROSS\_COMPILE=arm-himix200-linux- u-boot-z.bin** command.

The generated file **u-boot-hi3516cv500.bin** is the 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 HiTool 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 Hi3516C V500 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 *HiBurn User Guide*.

## 3.3 Burning the U-Boot to 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:

```
# mw.b <ddr_addr> ff 0x100000 /*Set the DDR value to all Fs.*/  
# tftp <ddr_addr> u-boot-hi3516cv500.bin /*Download the U-Boot to the DDR.*/  
# sf probe 0 /*Detect and initialize the SPI NOR flash.*/  
# sf erase 0x0 0x100000 /*Erase 1 MB capacity of the SPI NOR flash.*/  
# sf write <ddr_addr> 0x0 0x100000 /*Write the U-Boot from the DDR to the  
SPI NOR flash.*/
```



#### NOTE

The available value of **ddr\_addr** supported by Hi3516C V500 is **0x82000000**.

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

----End



## NOTICE

In the current version, you can perform block protection (BP) for the SPI NOR flash with the **sf lock** command. If a block of the SPI NOR Flash is protected, the block becomes read-only, and the erase and write commands do not take effect. In addition, the block protection does not lose effect even if the SPI NOR flash is powered off. Only the **sf lock 0** command can release the BP. For details, see the section [5.1.1 "BP 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:

```
# nand erase 0 0x100000 /*Erase 1 MB capacity of the NAND flash.*/  
# mw.b <ddr_addr> 0xff 0x100000 /*Set the DDR value to all Fs.*/  
# tftp <ddr_addr> u-boot-hi3516cv500.bin /*Download the U-Boot to the  
DDR.*/  
# nand write <ddr_addr> 0 0x100000 /*Write the U-Boot from the DDR to the  
NAND flash.*/
```



#### NOTE

The available value of **ddr\_addr** supported by Hi3516C V500 is **0x82000000**.

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

----End

### 3.4 Burning the U-Boot to the eMMC

Perform the following steps:

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

```
# mw.b <ddr_addr> 0xff 0x80000 /*Set the DDR value to all Fs.*/  
# tftp <ddr_addr> u-boot-hi3516cv500.bin /*Set the DDR value to all Fs.*/  
# mmc write 0 <ddr_addr> 0 0x400 /*Write the U-Boot from the DDR to the  
eMMC.*/
```



#### NOTE

The available value of **ddr\_addr** supported by Hi3516C V500 is **0x82000000**.

The format of the **mmc write** command is as follows:

```
mmc write <device num> addr blk# cnt
```

The parameters in the preceding command are described as follows:



- **<device num>**: device number
- **addr**: source address
- **blk#**: block serial number of the destination address
- **cnt**: number of the blocks. The block size is 512 bytes.

**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



### NOTE

In the following figures and descriptions, Hi35XX is used to replace Hi3516C V500.

## 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 connection to connect the DS-5 to the target platform by using the configuration database for the target platform.

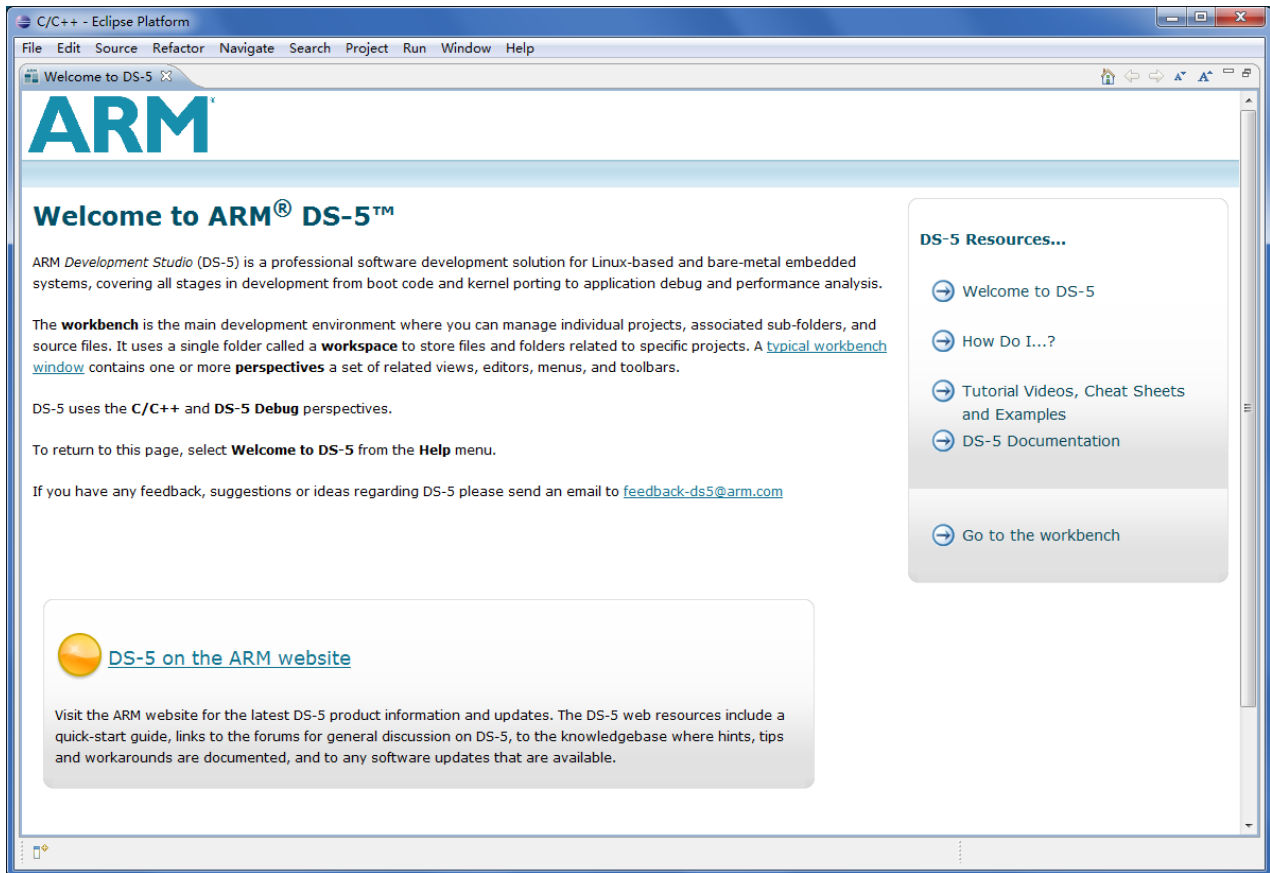
----End

### 4.3.1 Installing the ARM DS-5

The ARM 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



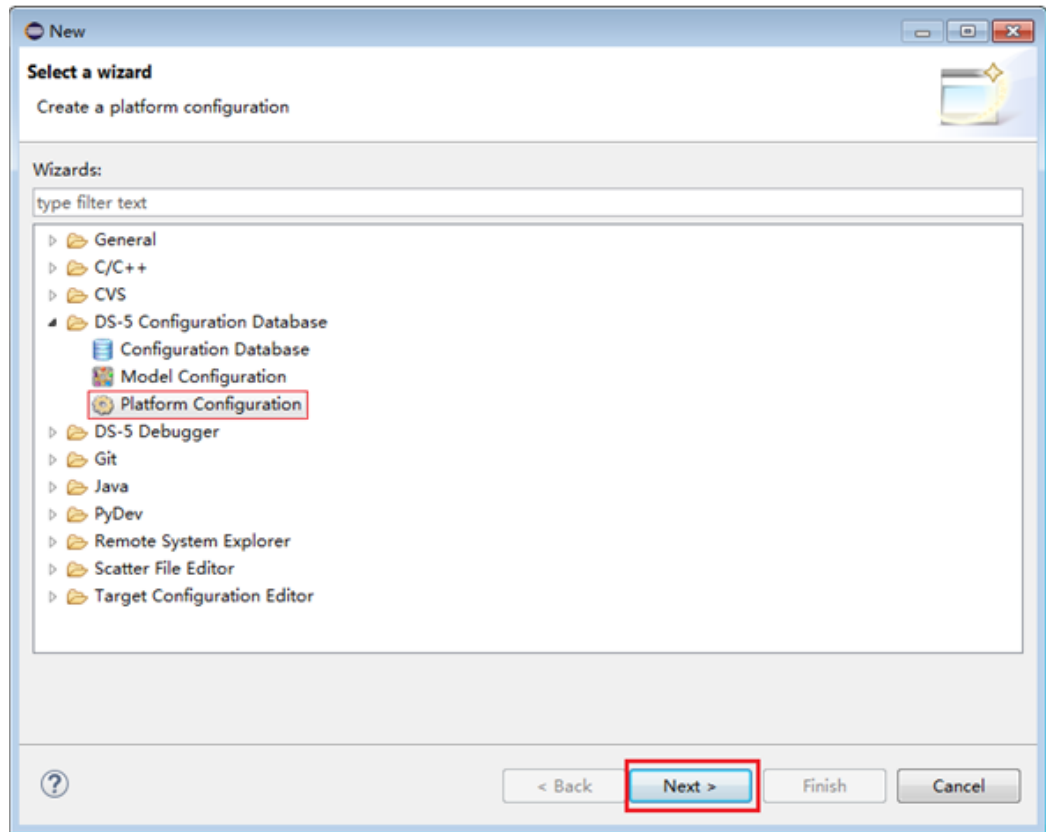
## 4.3.2 Creating a Configuration Database for the Target Platform

Perform the following steps:

- Step 1** Choose **File > New > Other**. In the displayed dialog box, select the **Platform Configuration** in the **DS-5 Configuration Database** folder, click **Next**, and configure the platform according to instructions.



**Figure 4-2** Platform configuration



**Step 2** Connect the simulator. Choose **Menu > ARM DS-5 v5.24.1 > Debug Hardware > Debug Hardware Config IP(5.24.1)** to enter the software interface, and click **Scan**. After the simulator appears, configure the IP address for the simulator to ensure that it is in the same network segment as the IP address for the PC.

**Figure 4-3** Config IP interface

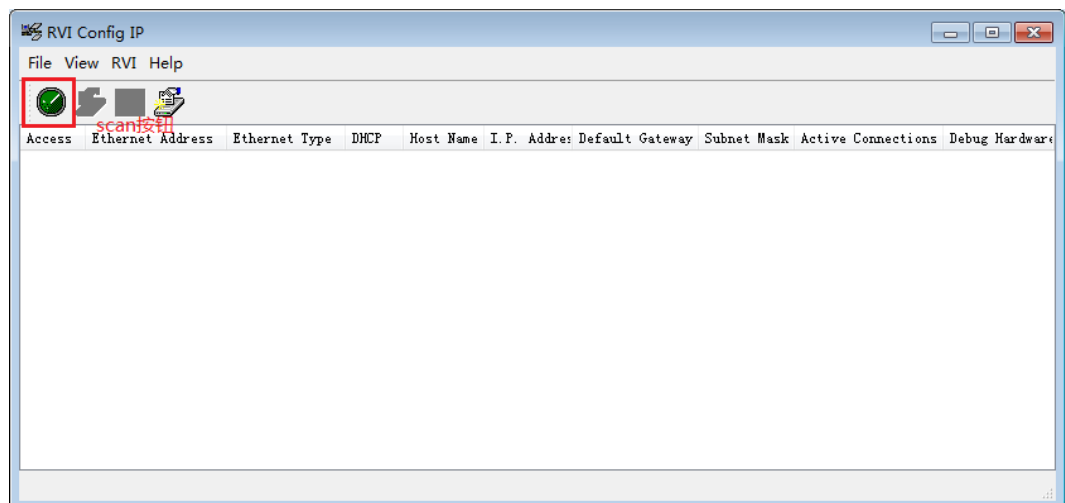




Figure 4-4 Config IP interface

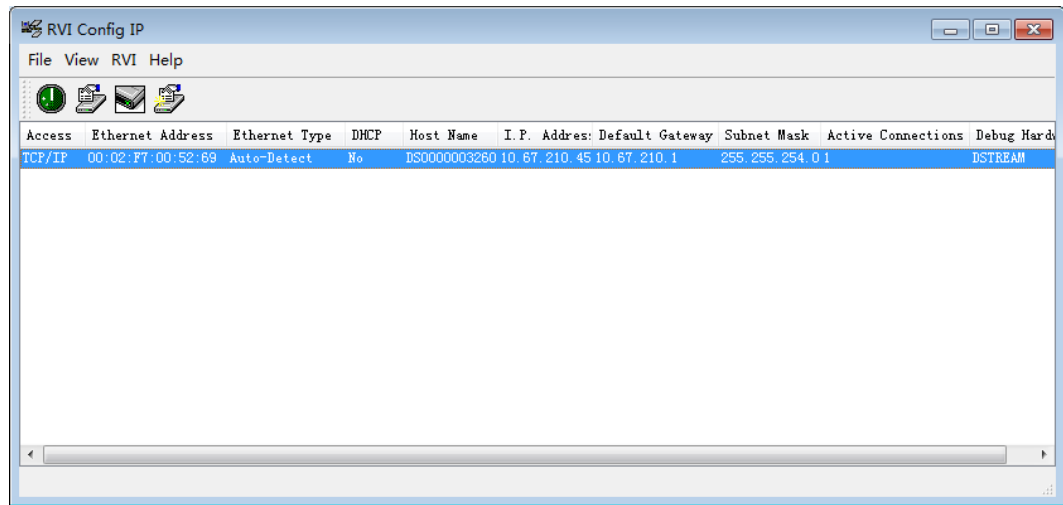
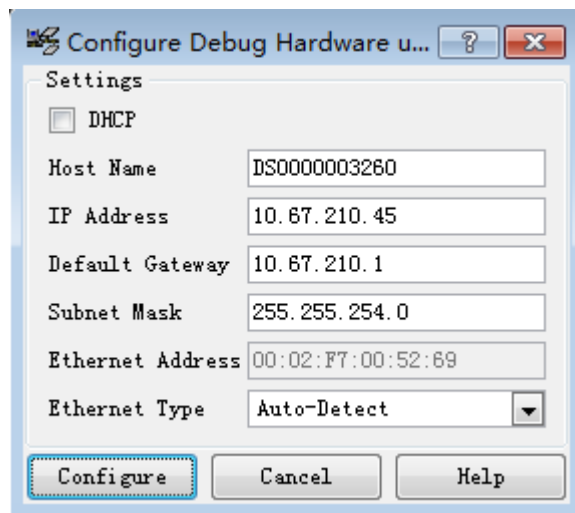


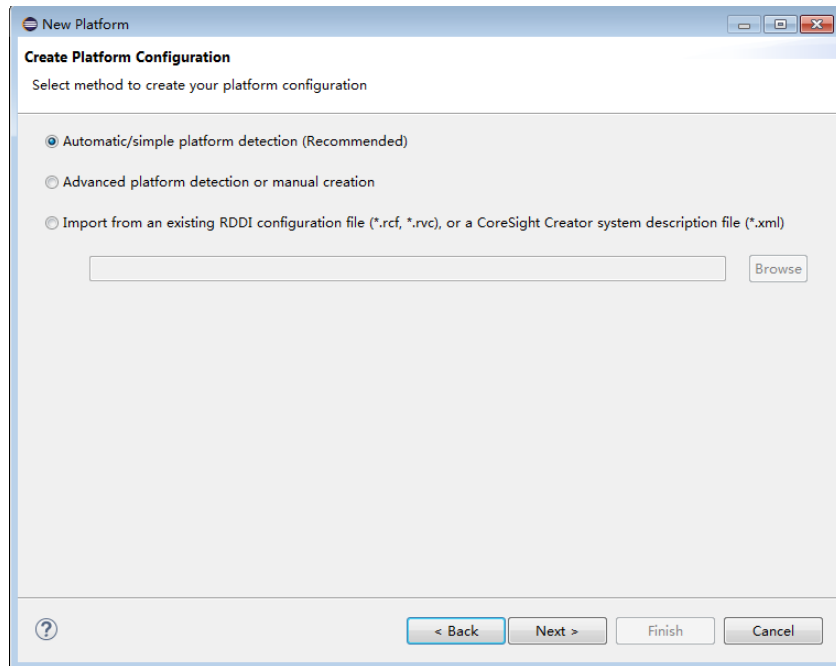
Figure 4-5 Simulator IP address configuration



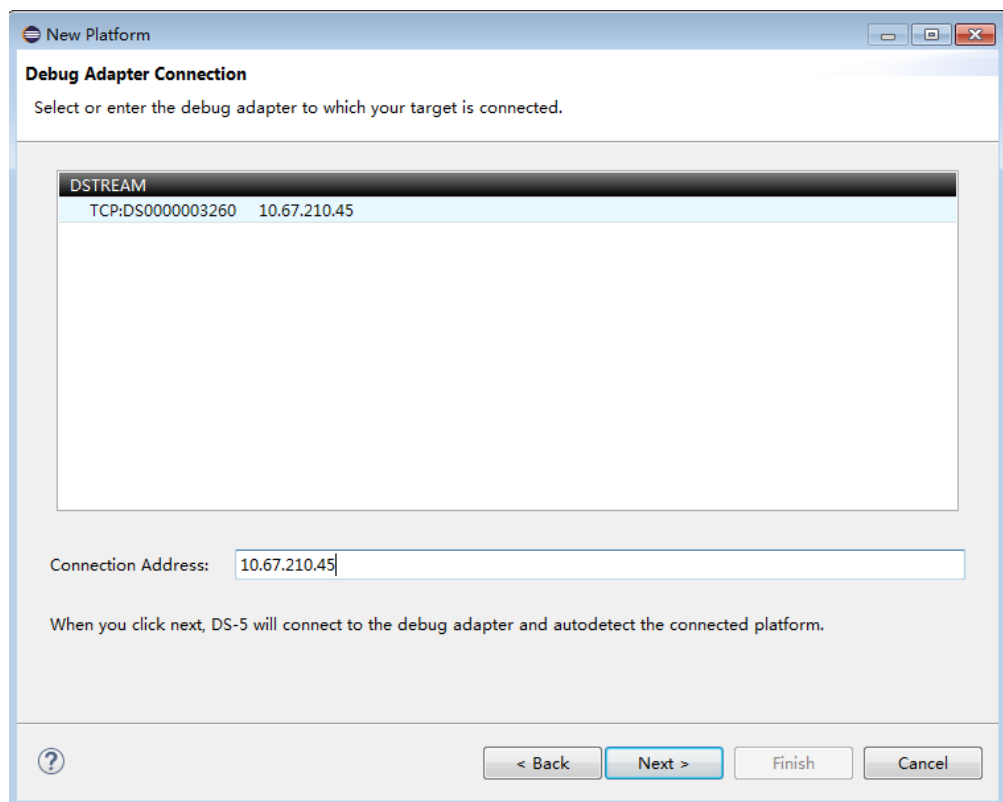
**Step 3** Go back to the **DS-5 Eclipse** page, select **Automatic/simple platform detection(Recommended)**, and click **Next**. The system automatically scans the simulator and enters the obtained IP address of the simulator to **Connection Address:**. Click **Next**, select **Debug target after saving configuration**, click **Next**, and then **Create New Database**. Enter the name and click **OK** and then **Next**. Change the content of **Platform Manufacturer** to **Hisilicon**, and change the value of **Platform Name** to **Hi35XX**. Click **Finish**. The configuration of the platform database is complete.



**Figure 4-6** Platform database configuration window 1—Create Platform Configuration

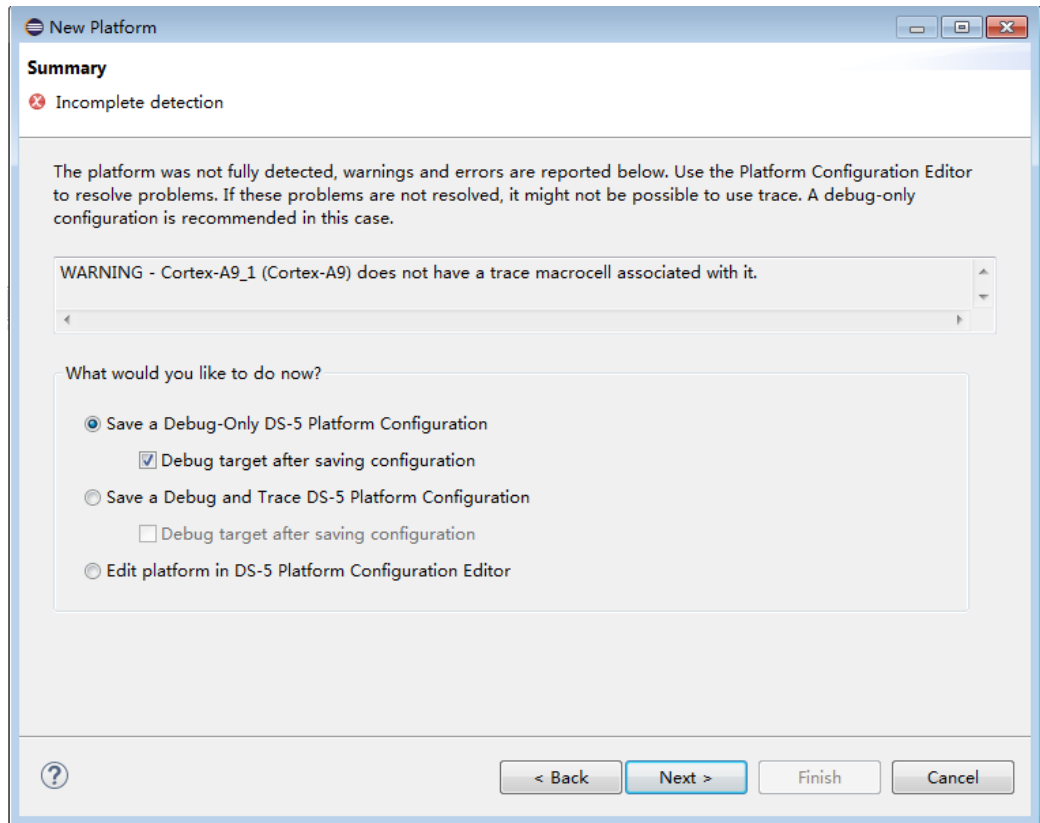


**Figure 4-7** Platform database configuration window 2—Debug Adapter Connection



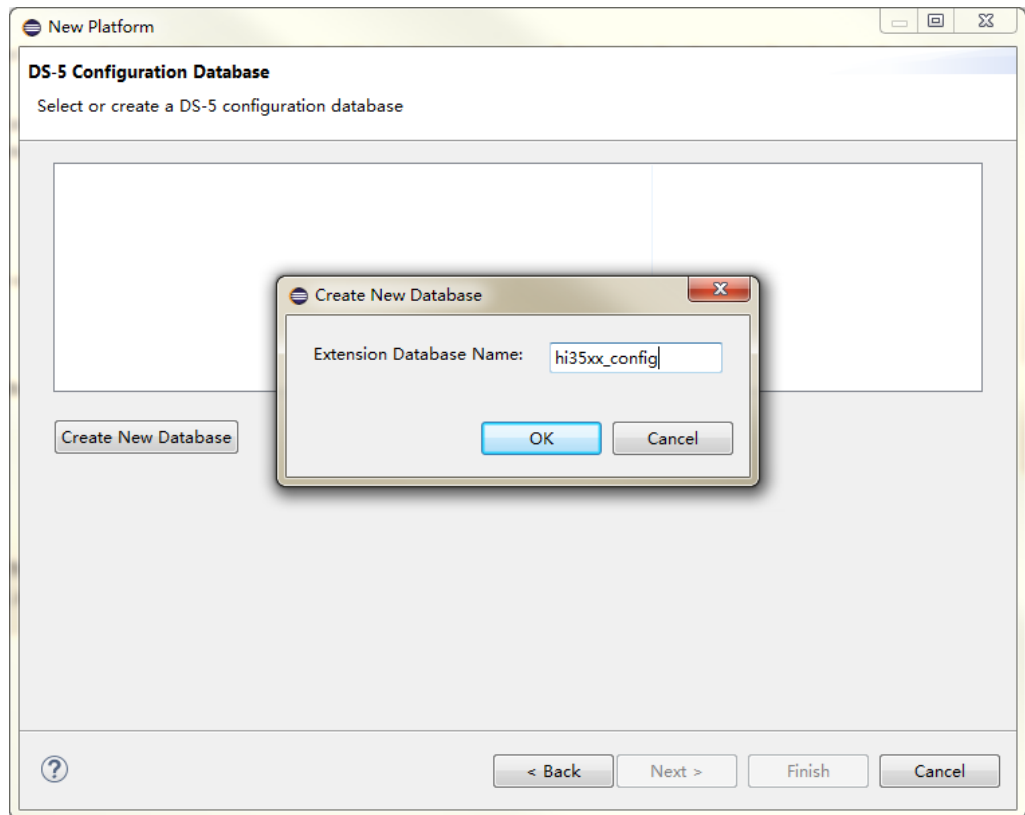


**Figure 4-8** Platform database configuration window 3—Summary





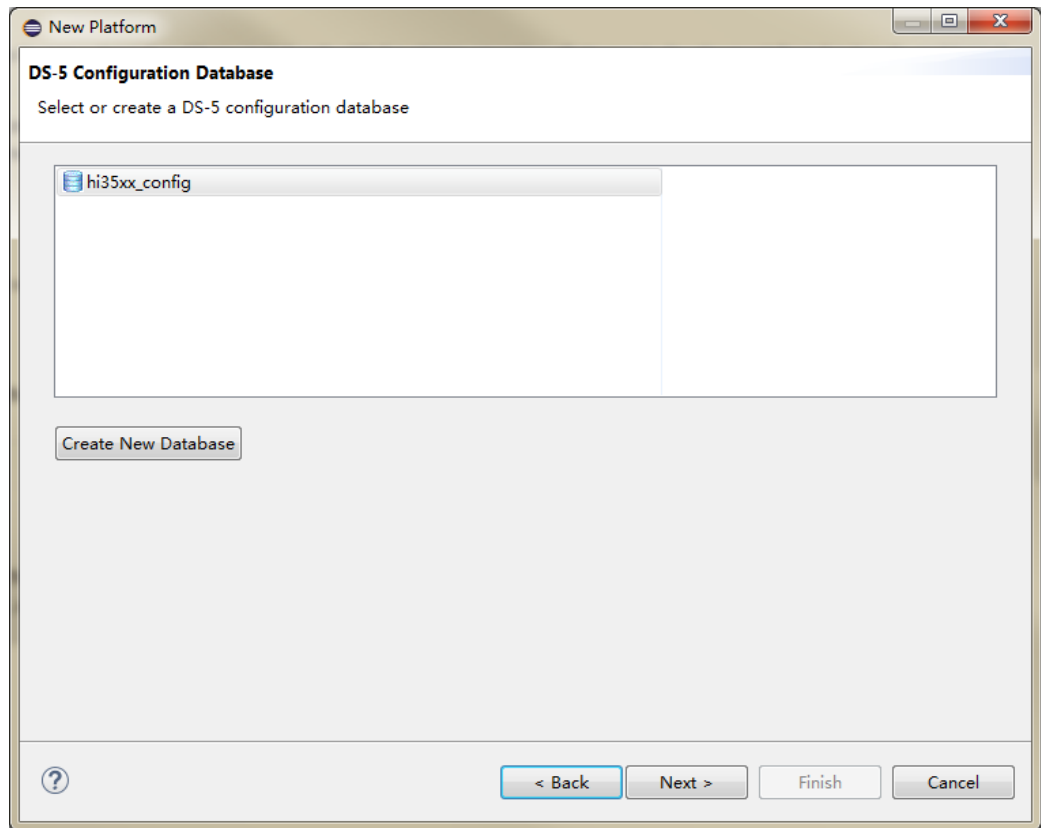
**Figure 4-9** Platform database configuration window 4—DS-5 Configuration Database—Create New Database







**Figure 4-10** Platform database configuration window 5—DS-5 Configuration Database—Create New Database





**Figure 4-11** Platform database configuration window 6—Platform Information

New Platform

**Platform Information**

Use this page to enter identification details for the platform.

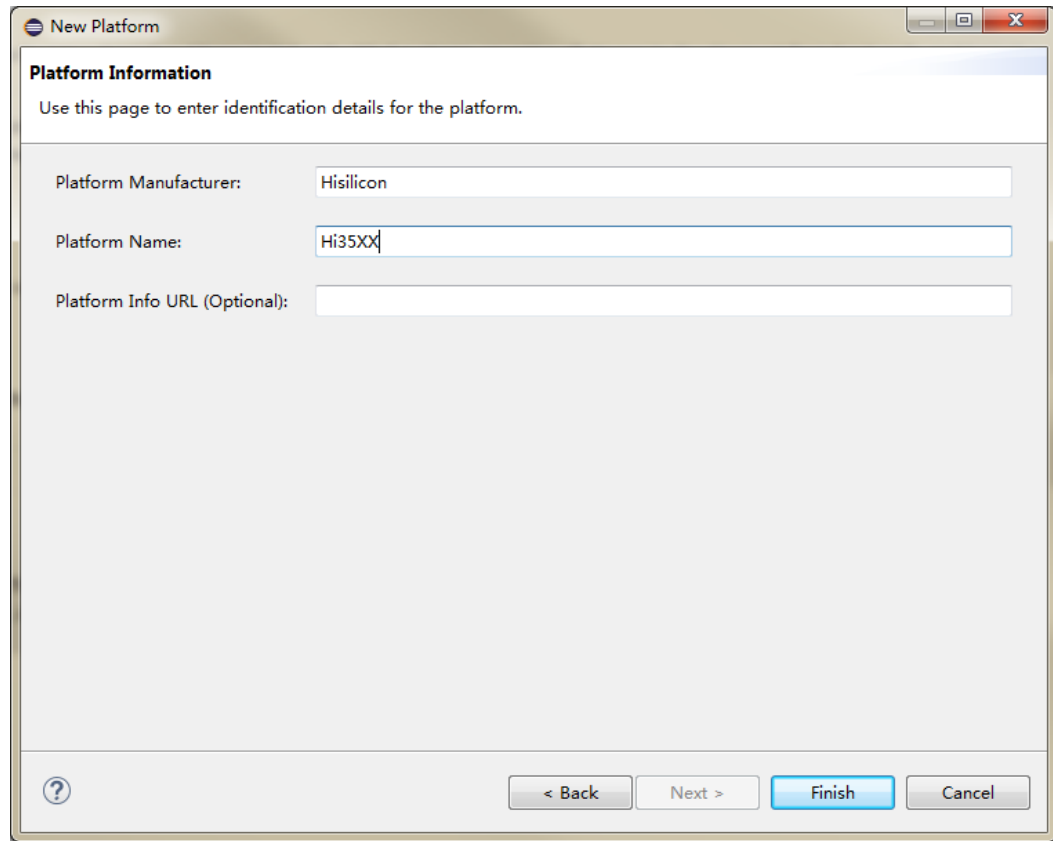
Platform Manufacturer: Imported

Platform Name: Imported Platform

Platform Info URL (Optional):

? < Back Next > Finish Cancel

**Figure 4-12** Platform database configuration window 7—Platform Information



----End

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

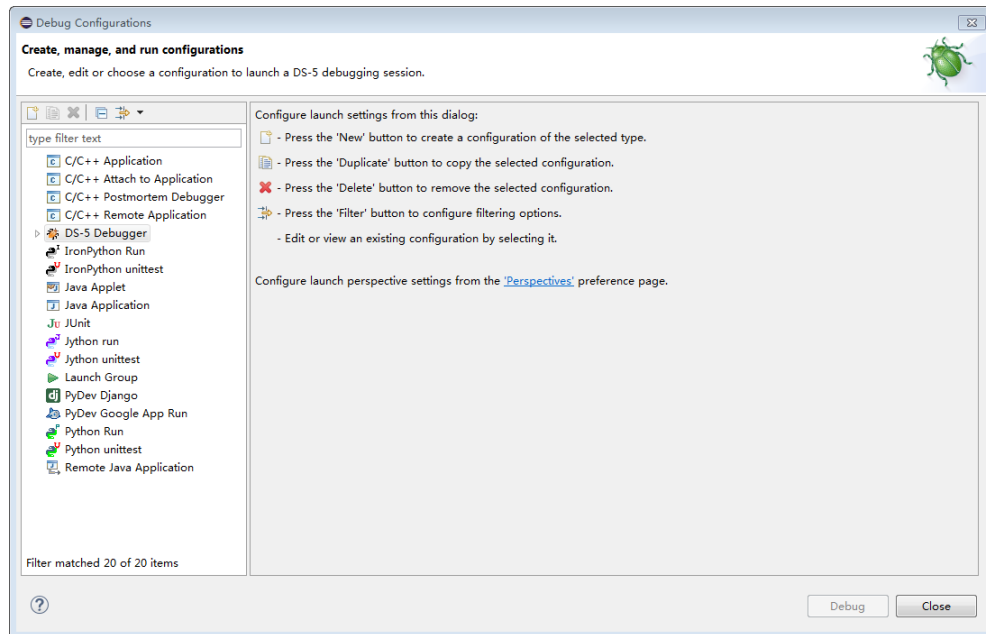
Perform the following steps:

- Step 1** A dialog box is displayed after **Finish** is clicked. In the name domain, find the DS-5 debugger, right-click **New**, and select the newly created **Hisilicon-Hi35XX**.
- Step 2** Choose **Hisilicon > Hi35XX > Bare Metal Debug > Cortex-A53** to select the added configuration database for the target platform in the **Connection** tab page, and enter the IP address of the DS-5 device in the text box. See [Figure 4-14](#).
- Step 3** Select **Connect Only** on the **Debugger** tab page. See [Figure 4-15](#).
- Step 4** Click **Debug** to connect the DS-5 to the target platform.

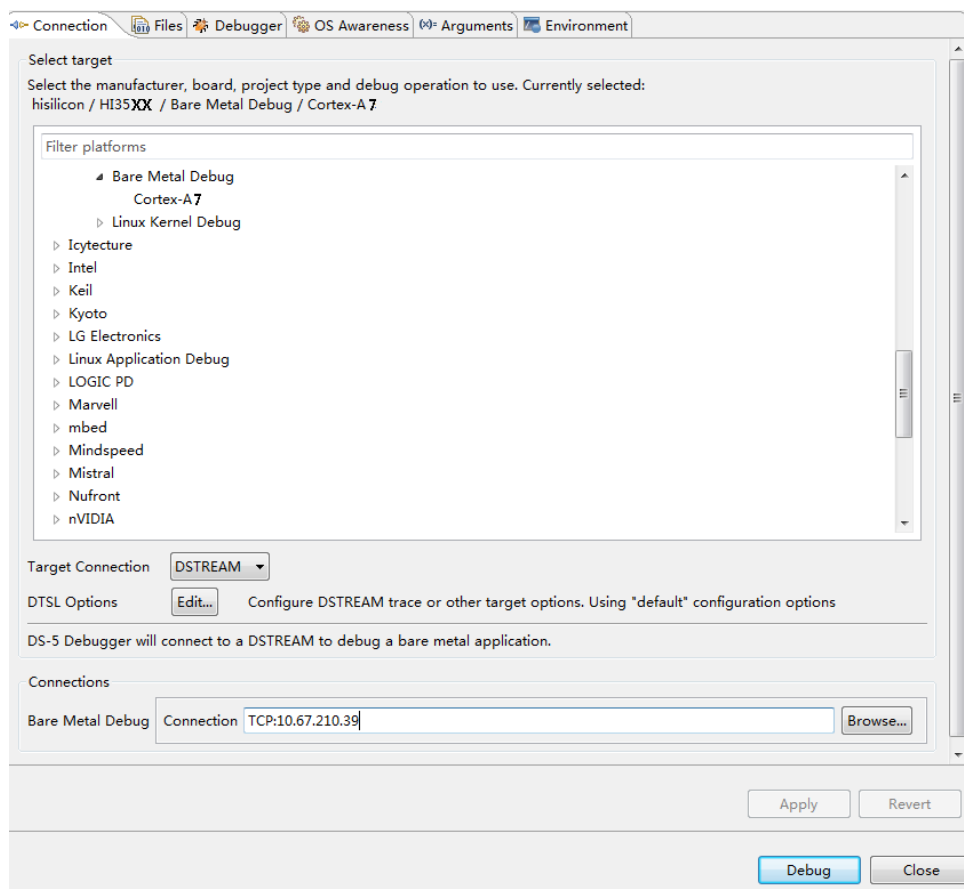
----End



**Figure 4-13** Debug Configurations window



**Figure 4-14** Debug Configurations window—selecting a new target platform configuration data base and entering the IP address of the DS-5 device





**Figure 4-15** Debug Configurations window—selecting Connect only

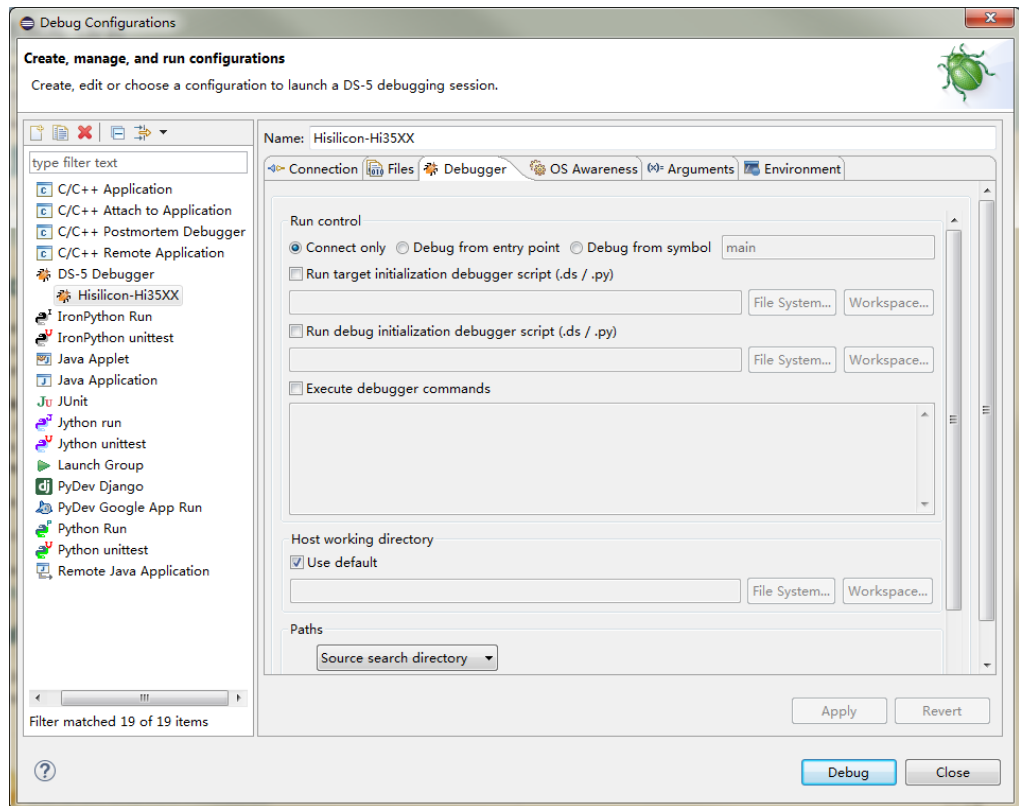
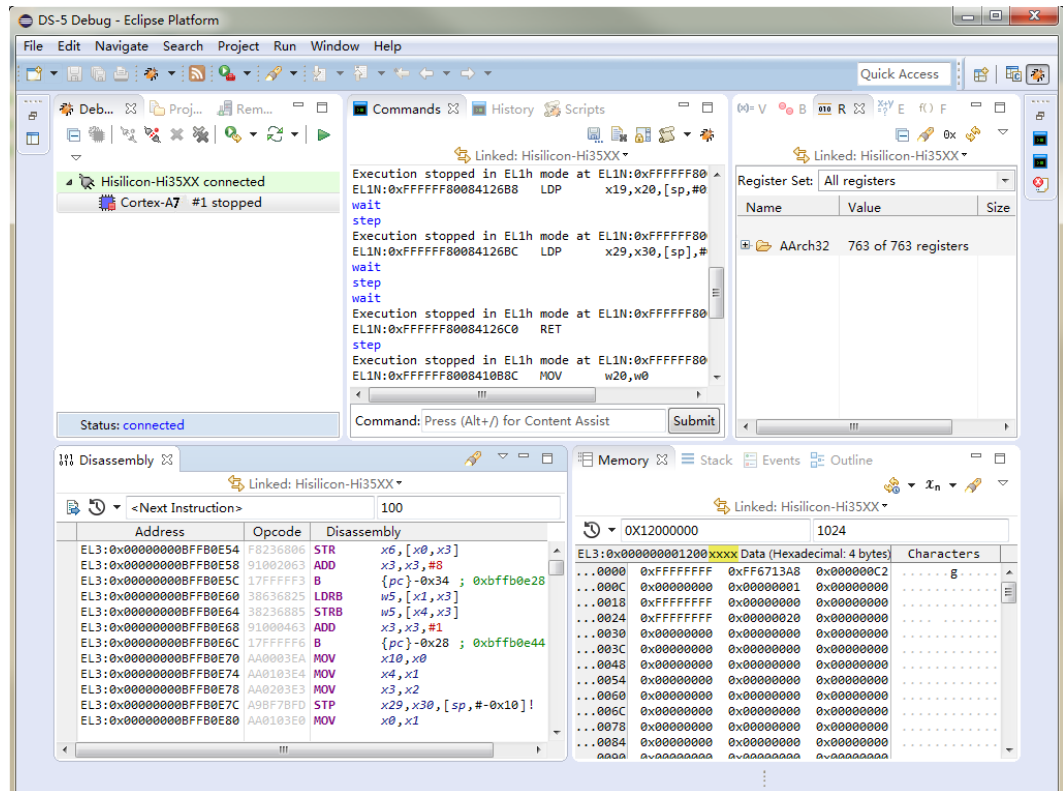







Figure 4-16 DS-5 Debug – Eclipse Platform window



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

### 4.4.1 Initializing the Memory

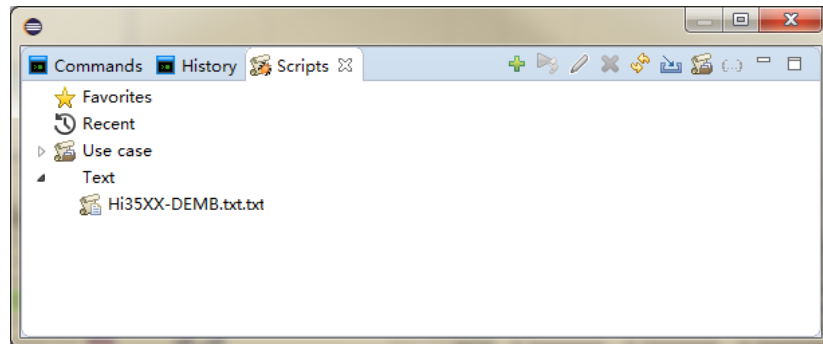
In the **Scripts** window, click  to import the memory initialization script, and click  to run the memory initialization script (if the simulator is running, click  in the **Debug Control** window to stop it). See [Figure 4-17](#).

#### NOTICE

The script for memory initialization is the file in .ds, .py or .txt format in the `osdrv/tools/pc/u-boot_tools` directory.



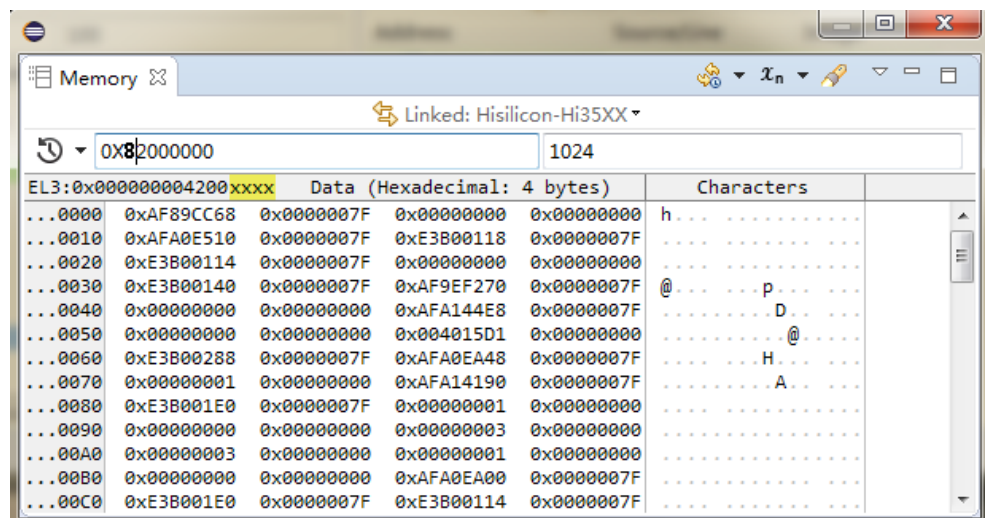
Figure 4-17 Scripts window



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 4-18. 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

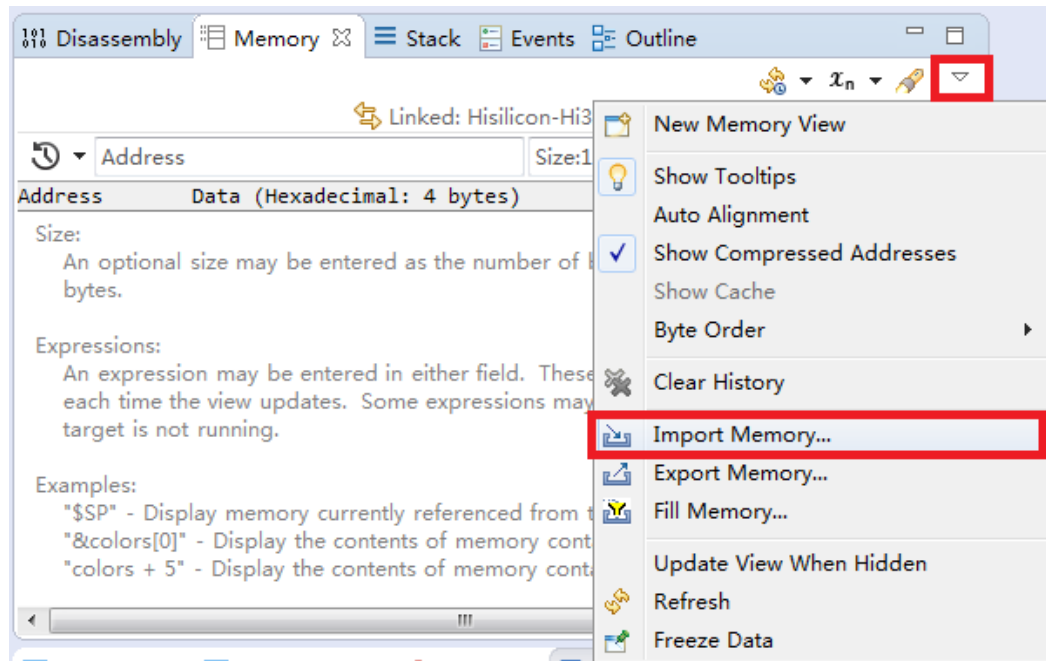


## 4.4.2 Downloading the U-Boot Image

Perform the following steps:

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

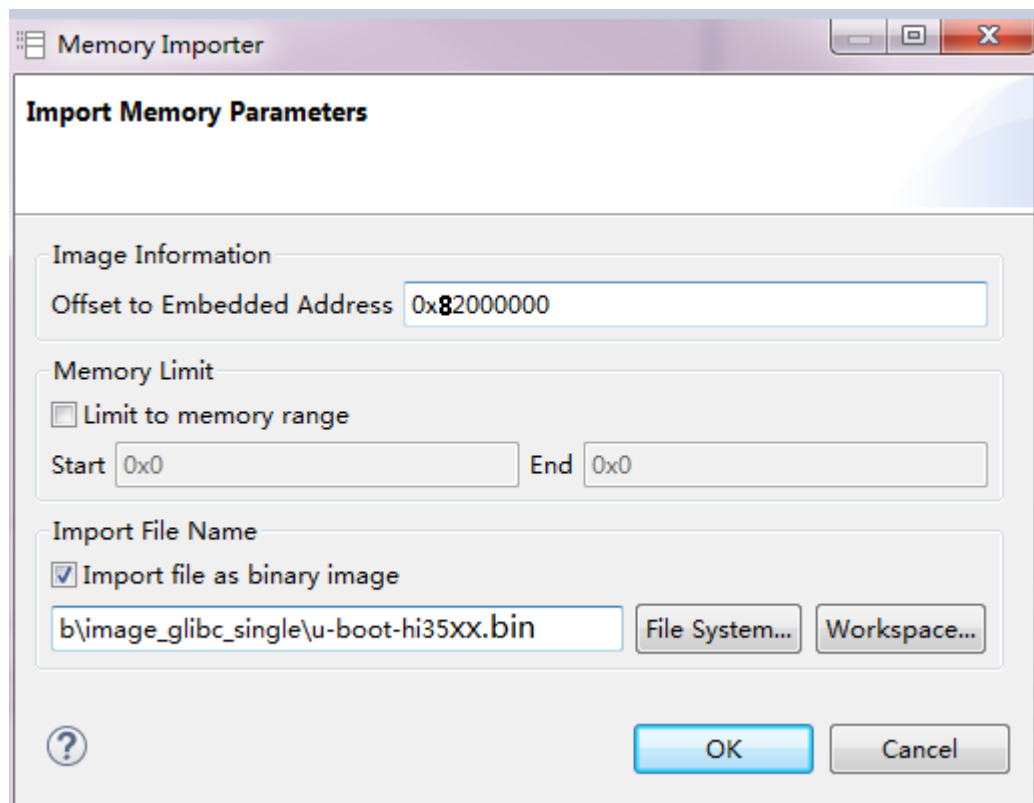
**Figure 4-19** Memory drop-down list



**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 4-20](#).

**Figure 4-20** Memory Import window

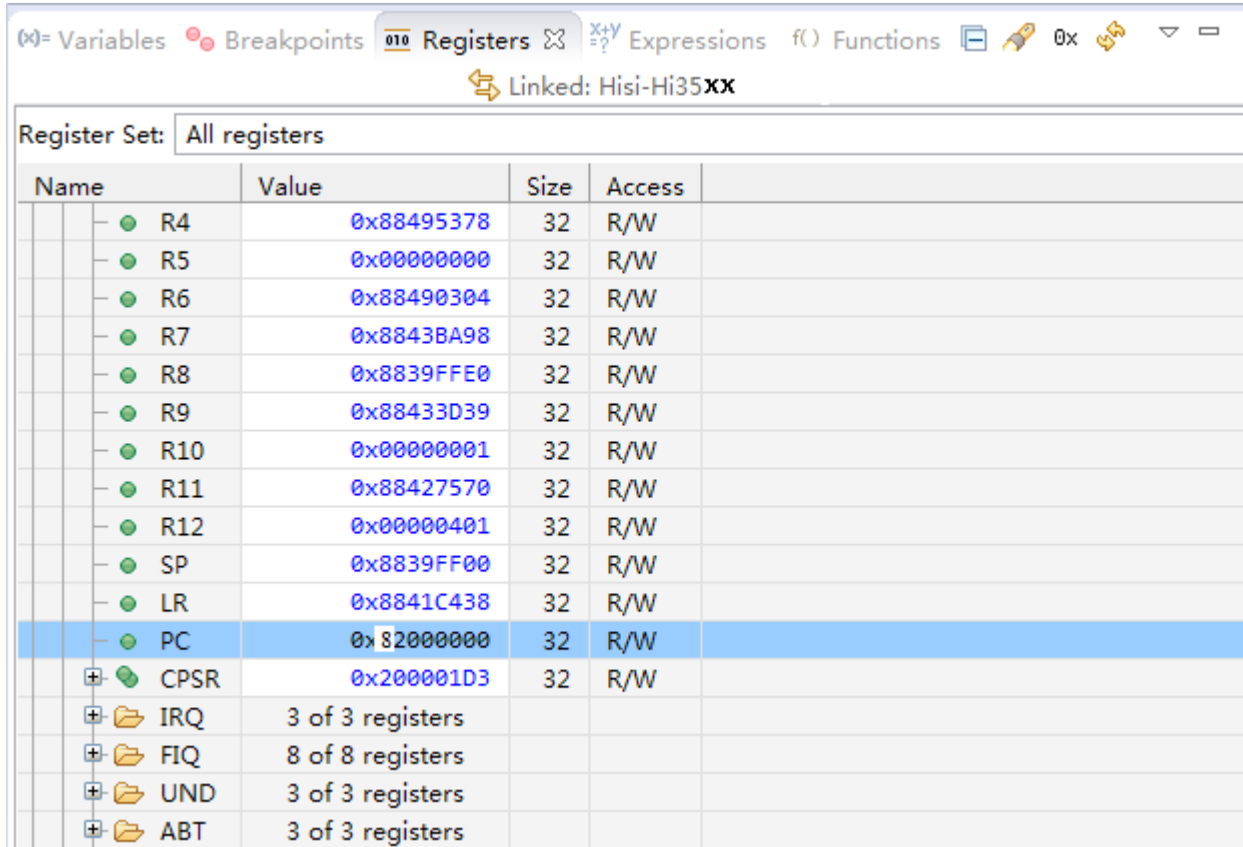







**Step 4** In the **Registers** window, change the PC value to **0x82000000**. See [Figure 4-21](#).

**Figure 4-21** Registers window



Name	Value	Size	Access
R4	0x88495378	32	R/W
R5	0x00000000	32	R/W
R6	0x88490304	32	R/W
R7	0x8843BA98	32	R/W
R8	0x8839FFE0	32	R/W
R9	0x88433D39	32	R/W
R10	0x00000001	32	R/W
R11	0x88427570	32	R/W
R12	0x00000401	32	R/W
SP	0x8839FF00	32	R/W
LR	0x8841C438	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		

**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 the boot media over the serial port.

Taking SPI NOR flash as an example, the burning commands are as follows:

```
# sf probe 0 /*Detect and initialize the SPI flash.*/  
# sf erase 0 0x100000 /*Erase 1 MB capacity of the SPI flash.*/  
# sf write <ddr_addr> 0 0x100000 /*Write the U-Boot from the memory to the  
SPI flash.*/  
# reset /*Restart the board*/
```



**NOTE**

The available value of **ddr\_addr** supported by Hi3516C V500 is **0x82000000**.



# 5 Appendix

## 5.1 U-Boot Command Description

### 5.1.1 BP Command for the SPI NOR Flash

Common SPI NOR flash components provide the BP bit to for data protection.

You can set the BP bits (BP0, BP1, BP2, BP3, and BP4) in the status register (SR) to write-protect certain blocks. (BP3 and BP4 are not provided by all vendors.) The BP bits are nonvolatile bits, that is, as long as the BP bits are set, the blocks are write-protected even if the SPI NOR flash component is powered off.

Some SPI NOR flash components also provide the top/bottom block protection (TBPROT) bit in the configuration register (CR). This bit can be used to configure whether the BP starts at top (the high address) or at bottom (the low address). This bit is one-time programmable (OTP). The default value is **0**, indicating that the BP starts at top (the high address). Once this bit is changed to **1**, the BP starts at bottom (the low address) and cannot be changed anymore.

The TBPROT bit is set to **1** for application purpose in this version, that is, the BP protection starts at bottom (the low address).

The default values of the BP bits in the SPI NOR flash SR are **0** (BP disabled), that is, all blocks on the component are unprotected, and are writable and erasable.

You can set all the BP bits to **1** (BP enabled) to protect all blocks on the component from write or erase.

The BP is implemented based on the unit of block. According to the status of the BP bits, the values are converted into base-10 level values to determine the BP area. For components with three BP bits, the level range is 0 to 7 between BP[0:0:0] and BP[1:1:1]. For components with four BP bits, the level range is 0 to 10 (or 0 to 9, because the minimum BP lock area is one block) between BP[0:0:0:0] and BP[1:1:1:1].

The BP area varies according to different components from different vendors, as shown in [Table 5-1](#).



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

Level	MXIC MX25L-					ESMT F25L-	
	12835F	25635F	25735F	6406E	1606E	64QA	
0	Unlock	Unlock	Unlock	Unlock	Unlock	Unlock	
1	0–64 KB	0–64KB	0–64 KB	0–4 MB (64 blocks)	Lock all 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	Lock all 8 MB (128 blocks)	Lock all 2 MB (32 blocks)	Lock all 8 MB (128 blocks)	
8	0–8 MB	0–8 MB	0–8 MB	-	-	-	
9	Lock all 16 MB	0–16 MB	0–16 MB	-	-	-	
10	-	Lock all 32 MB	Lock all 32 MB	-	-	-	
Level	SPANSION S25FL-			WINBOND W25Q-			
	127S	256S		128FV	128BV	256FV	64FV
0	Unlock	Unlock		Unlock	Unlock	Unlock	Unlock
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	Lock all 16 MB	Lock all 32 MB	Lock all 16 MB	Lock all 16 MB	0-4 MB	Lock all 8 MB
8	-	-	-	-	0-8 MB	-
9	-	-	-	-	0-16 MB	-
10	-	-	-	-	Lock all 32 MB	-
Level	GD GD25Q-			CFEON EN25Q-		
	128C	64	32	128	64	
0	Unlock	Unlock	Unlock	Unlock	Unlock	
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	Lock all 16 MB	Lock all 8 MB	Lock all 4 MB	Lock all 16 MB (256 blocks)	Lock all 8 MB (128 blocks)	

A series of lock commands are added for the block protection function under the U-Boot:

- **sf lock**

This command is used to check the current lock level, level range, and the locked area. The command description is also printed, as shown in [Figure 5-1](#).

**Figure 5-1** Checking the 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 blocks on the component by setting the maximum lock level, as shown in [Figure 5-2](#). For details of the level information, see [Table 5-1](#).

**Figure 5-2** Locking all blocks

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

- **sf lock 0**

This command is used to unlock the BP for all blocks on the component, as shown in [Figure 5-3](#). Then, you can write data to or erase data from any block.

**Figure 5-3** Unlocking all blocks

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

- **sf lock <level>**

This command is used to specify a BP area by setting the lock level, as described in [Table 5-1](#). In this way, the blocks in the BP area are not writable or erasable. See [Figure 5-4](#).

**Figure 5-4** Specifying a BP area by setting the lock level

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