



Hi3519AV100/Hi3556AV100 U-boot 移植应用

开发指南

文档版本 00B03

发布日期 2018-07-30

版权所有 © 深圳市海思半导体有限公司 2018。保留一切权利。

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前言

概述

本文档主要介绍在 Hi3519AV100/Hi3556AV100 单板上如何移植和烧写 U-boot (Hi3519AV100/Hi3556AV100 单板的 Bootloader) 的相关操作及如何使用 ARM 调试工具。



说明

本文以 Hi3519AV100 描述为例，未有特殊说明，Hi3556AV100 与 Hi3519AV100 一致。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3519A	V100
Hi3556A	V100

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。



修订日期	版本	修订说明
2018-07-30	00B03	添加 Hi3556AV100 的相关内容
2018-07-06	00B02	第 2 次临时版本发布 2.2、2.5、3.3 和 4.4.3 小节涉及修改
2018-05-15	00B01	第 1 次临时版本发布



目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 U-boot 目录结构	1
2 移植 U-boot.....	3
2.1 U-boot 硬件环境	3
2.2 编译 U-boot	3
2.3 配置 DDR 存储器	4
2.4 配置管脚复用	4
2.5 生成最终使用的 U-boot 镜像.....	4
3 烧写 U-boot.....	5
3.1 概述.....	5
3.2 通过 bootrom 工具烧写 U-boot	5
3.3 Flash 的 U-boot 烧写方法.....	5
3.3.1 SPI -Nor Flash 烧写方法.....	5
3.3.2 NAND Flash/SPI-Nand Flash 烧写方法	6
3.4 eMMC 的 U-boot 烧写方法	6
4 如何使用 ARM 调试工具	8
4.1 概述.....	8
4.2 ARM 调试工具简介	8
4.2.1 DS-5 Eclipse	8
4.2.2 DS-5 Debug	9
4.3 使用 ARM 调试工具.....	9
4.3.1 安装 ARM Development Studio 5	9
4.3.2 新建目标平台配置数据库.....	10
4.3.3 连接目标平台.....	18
4.4 使用仿真器烧写 Flash	22
4.4.1 内存初始化	22
4.4.2 下载 U-Boot 映像.....	23



4.4.3 烧写映像	26
5 附录.....	27
5.1 u-boot 命令说明	27
5.1.1 SPI-Nor 块保护命令	27



插图目录

图 4-1 DS-5 Eclipse 启动界面	10
图 4-2 平台配置界面	11
图 4-3 Config IP 界面	11
图 4-4 Config IP 扫描界面	12
图 4-5 配置仿真器 IP 界面	12
图 4-6 配置平台数据库界面—Create Platform Configuration	13
图 4-7 配置平台数据库界面—Debug Adapter Connection	13
图 4-8 配置平台数据库界面—Summary	14
图 4-9 配置平台数据库界面—DS-5 Configuration Database—Create New Database	15
图 4-10 配置平台数据库界面—DS-5 Configuration Database—完成 “Create New Database”	16
图 4-11 配置平台数据库界面—Platform Information	17
图 4-12 配置平台数据库界面—完成 “Platform Information” 配置	18
图 4-13 Debug Configurations 窗口	19
图 4-14 Debug Configurations 窗口—选择新添加的目标平台配置数据库并输入 DS-5 设备的 IP 地址	20
图 4-15 Debug Configurations 窗口—勾选 “Connect only”	21
图 4-16 DS-5 Debug – Eclipse Platform 窗口	22
图 4-17 脚本窗口	23
图 4-18 Memory 窗口	23
图 4-19 Memory 下拉窗口	24
图 4-20 Memory Importer 窗口	25
图 4-21 Registers 窗口	26
图 5-1 查看当前块保护信息	29
图 5-2 锁定整个器件	30
图 5-3 解除当前锁定状态	30
图 5-4 通过设置 level 值锁定指定区域	30



表格目录

表 1-1 U-boot 的主要目录结构	1
表 5-1 不同厂家不同器件的块保护锁定区域与 BP Level 对应表	28



1 概述

1.1 概述

Hi3519AV100 单板的 Bootloader 采用 U-boot。当选用的外围芯片的型号与单板上外围芯片的型号不同时，需要修改 U-boot 配置文件，主要包括存储器配置、管脚复用。

1.2 U-boot 目录结构

U-boot 的主要目录结构如表 1-1 所示，详细目录说明请阅读 U-boot 目录下的 README 文档。

表1-1 U-boot 的主要目录结构

目录名	描述
arch	各种芯片架构的相关代码、U-boot 入口代码。
board	各种单板的相关代码，主要包括存储器驱动等。
board/hisilicon/hi3519av100	Hi3519AV100 单板相关代码。
arch/xxx/lib	各种体系结构的相关代码，如 ARM、MIPS 的通用代码。
include	头文件。
include/configs	各种单板的配置文件。
common	各种功能（命令）实现文件。
drivers	网口、Flash、串口等的驱动代码。
net	网络协议实现文件。
fs	文件系统实现文件。
product/hiupdate	SD 卡升级、USB 升级功能实现



目录名	描述
product/hiosd	dec、hdmi 接口、vo、mipi 功能实现
product/hiaudio	音频功能实现



2 移植 U-boot

2.1 U-boot 硬件环境

Hi3519AV100 DMEB 板上的外围芯片包括 DDR SDRAM、eMMC、NAND Flash、SPI Nor Flash 和 SPI-NAND Flash，其具体型号见《Hi3519AV100 Compatibility Test Report.xlsm》。

2.2 编译 U-boot

当所有移植步骤完成后，就可以编译 U-boot，操作如下：

步骤 1. 配置编译环境

使用配置命令：

```
make ARCH=arm CROSS_COMPILE=arm-himixXXX-linux- hi3519av100_config
```



说明

Hi3519AV100 和 Hi3556AV100 对应的配置文件不同，使用中请根据平台选择对应的配置文件。

步骤 2. 编译 U-boot

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

编译成功后，将在 U-boot 目录下生成 u-boot.bin。



说明

编译 uboot 时需要在 make 后添加两个参数：ARCH=arm CROSS_COMPILE=arm-himixXXX-linux-，其中 CROSS_COMPILE 表示工具链。

- 使用 uclibc 工具链时，CROSS_COMPILE= arm-himix100-linux-
- 使用 glibc 工具链时，CROSS_COMPILE=arm- himix200-linux-
- 本文档中统一以“arm-himixXXX-linux”来表示这两种情况，后续不再重复说明。

---结束



注意

这一步生成的 u-boot.bin 只是一个中间件，并不是最终在单板上执行的 U-boot 镜像。

2.3 配置 DDR 存储器

在 Windows 下打开 SDK 中的“osdrv/tools/pc/u-boot_tools/”目录下的配置表格。当选用不同的 DDR SDRAM 时，需要针对不同器件的特性，对配置工作表中的 DDR 相关标签页进行修改。

2.4 配置管脚复用

如果管脚复用有变化，还需要对配置表格中的管脚复用相关标签页进行修改。

2.5 生成最终使用的 U-boot 镜像

U-boot 镜像生成步骤如下：

- 步骤 1. 完成配置表格的修改后，保存表格。
- 步骤 2. 单击表格第一个标签页上的按钮【Generate reg bin file】，生成临时文件 reg_info.bin。
- 步骤 3. 将生成的 u-boot.bin 复制到 osdrv/tools/pc/u-boot_tools/目录

```
./mkboot.sh reg_info.bin u-boot-hi3519av100.bin
```

生成的 u-boot-hi3519av100.bin 就是能够在单板上运行的 uboot 镜像。

----结束



3 烧写 U-boot

3.1 概述

如果待移植单板中已有 U-boot 运行，则可以通过串口或网口与服务器连接，直接更新 U-boot。

如果是第一次烧写，则需要使用 HiTool 或者 DS-5 工具进行烧写。由于芯片特性，在使用 DS-5 时必须要对存储器和芯片进行初始化。在 Hi3519AV100 SDK 中提供了相应的初始化脚本，当选用了不同的外围芯片，则需要重新配置初始化脚本才能使用。

3.2 通过 bootrom 工具烧写 U-boot

具体操作方式请参考《HiBurn 工具使用说明》。

3.3 Flash 的 U-boot 烧写方法

3.3.1 SPI -Nor Flash 烧写方法

SPI-Nor Flash 烧写方法如下：

步骤 1. 在内存中运行起来之后在超级终端中输入：

```
# mw.b <ddr_addr> ff 0x100000 /* 对内存初始化*/  
# tftp <ddr_addr> u-boot-hi3519av100.bin /*U-boot下载到内存*/  
# sf probe 0 /*探测并初始化SPI-Nor flash*/  
# sf erase 0x0 0x100000 /*擦除 1M大小*/  
# sf write <ddr_addr> 0x0 0x100000 /*从内存写入SPI-Nor Flash*/
```



说明

Hi3519AV100 平台的<ddr_addr>可用地址 0x22000000。

步骤 2. 上述步骤操作完成后，重启系统可以看到 U-boot 烧写成功。

----结束



注意

在当前版本，使用 `sf lock` 可以对 SPI Nor Flash 进行块保护（Blocks Protect）。如果对 SPI Nor Flash 的某个块进行了块保护，这个块就变成只读，运行擦除和写命令都不会生效，而且掉电并不能失效块保护。在这种情况下，只有在执行 `sf lock 0` 命令，解除块保护之后，SPI Nor Flash 擦除和写操作才会起作用。详见 5.1.1 章节“SPI-Nor 块保护命令”。

3.3.2 NAND Flash/SPI-Nand Flash 烧写方法

NAND Flash/SPI-Nand Flash 烧写方法如下：

步骤 1. 在内存中运行起来之后在超级终端中输入：

```
# nand erase 0 0x100000          /*擦除 1M大小*/  
# mw.b <ddr_addr> 0xff 0x100000 /* 对内存初始化*/  
# tftp <ddr_addr> u-boot-hi3519av100.bin /*U-boot下载到内存*/  
# nand write <ddr_addr> 0 0x100000 /*从内存写入NAND Flash*/
```



说明

Hi3519AV100 平台的<ddr_addr>可用地址 0x22000000。

步骤 2. 重启系统可以看到 U-boot 烧写成功。

----结束



注意

烧写 NAND Flash 和 SPI-NAND Flash 使用同样 `nand` 命令，因此要求单板上两种 NAND Flash 不能共存。

3.4 eMMC 的 U-boot 烧写方法

eMMC 烧写方法如下：

步骤 1. 在内存中运行起来之后在超级终端中输入：

```
# mw.b <ddr_addr> 0xff 0x80000    /* 对内存初始化*/  
# tftp <ddr_addr> u-boot-hi3519av100.bin /* U-boot下载到内存*/  
# mmc write 0 <ddr_addr> 0 0x400 /*从内存写入eMMC*/
```



说明

- Hi3519AV100 平台的<ddr_addr>可用地址 0x22000000。
- mmc write 命令说明：
格式：mmc write <device num> addr blk# cnt
- 参数：
<device num>: 设备号
addr: 源地址
blk#: 目的地址的块序号
cnt: 块的数目，块大小为 512 字节

步骤 2. 重启系统可以看到 U-boot 烧写成功。

----结束



4 如何使用 ARM 调试工具

4.1 概述

DS-5，即 ARM Development Studio 5，是一款针对 ARM 支持的 Linux 和 Android 平台的全面的端到端软件开发工具套件，内容涵盖启动代码和内核移植以及应用程序和裸机调试各个阶段的开发。

ARM DS-5 提供具有跟踪、系统范围性能分析器、实时系统模拟器和编译器的应用程序和内核空间调试器。这些功能包括在定制、功能强大且用户友好的基于 Eclipse 的 IDE 中。借助于该工具套件，可以很轻松地，为 ARM 支持的平台开发和优化基于 Linux 的系统，缩短开发和测试周期，并且可帮助工程师创建资源利用效率高的软件。

DS-5 主要包括：

- DS-5 Eclipse：集成开发环境（IDE），将编译和调试工具结合在一起。
- DS-5 Debug。
- Real-Time System Models（RTSM）：实时系统模型。
- ARM 流水线性能分析器。

本章介绍了关于 ARM 处理器调试用到的调试工具的使用方法，调试工具包括：

- DS-5 Eclipse
- DS-5 Debug



说明

下文截图和描述中的平台名称统一采用 Hi35XX 替代 Hi3519AV100。

4.2 ARM 调试工具简介

4.2.1 DS-5 Eclipse

DS-5 Eclipse 是一种集成开发环境（IDE），该集成环境在 Eclipse 基础上集成了 ARM 的编译和调试工具，以及针对 ARM Linux 目标板开发的 ARM Linux GNU 工具链。

DS-5 Eclipse 包括项目管理、编辑器和视图等主要功能。



4.2.2 DS-5 Debug

DS-5 Debug 是一个图形化调试器，支持在 ARM 目标板和 Real-Time System Models (RTSM) 上直接进行软件开发调试。全面和直观的视图非常易于调试 Linux 和裸机程序，包括源程序同步和反汇编，堆栈调用管理，内存、寄存器、表达式、变量、线程和断点操作，以及代码跟踪。

使用 Debug 管理窗口，可以在源码级或指令级单步执行，并在其他视图中查看代码执行后的最新数据。也可以设置断点或观察点暂停程序继续执行，以便了解应用程序执行后的行为。在一些目标板上还可以使用跟踪视图，以程序运行的先后顺序跟踪应用程序中函数的执行。

4.3 使用 ARM 调试工具

要使用 DS-5 进行程序调试或者向裸板烧写 U-boot 程序，首先必须创建目标平台配置数据库，然后才能连接到目标平台进行程序调试或者向开发板烧写 U-boot 程序。

关于使用 ARM 调试工具的更详细描述请参见 ARM 公司提供的文档。下面介绍如何使用 DS-5。

步骤 1. 安装 ARM Development Studio 5。

步骤 2. 创建目标平台配置数据库。

步骤 3. 连接到目标平台。创建一个新的连接，使用该目标平台配置数据库将 DS-5 设备连接到目标平台。

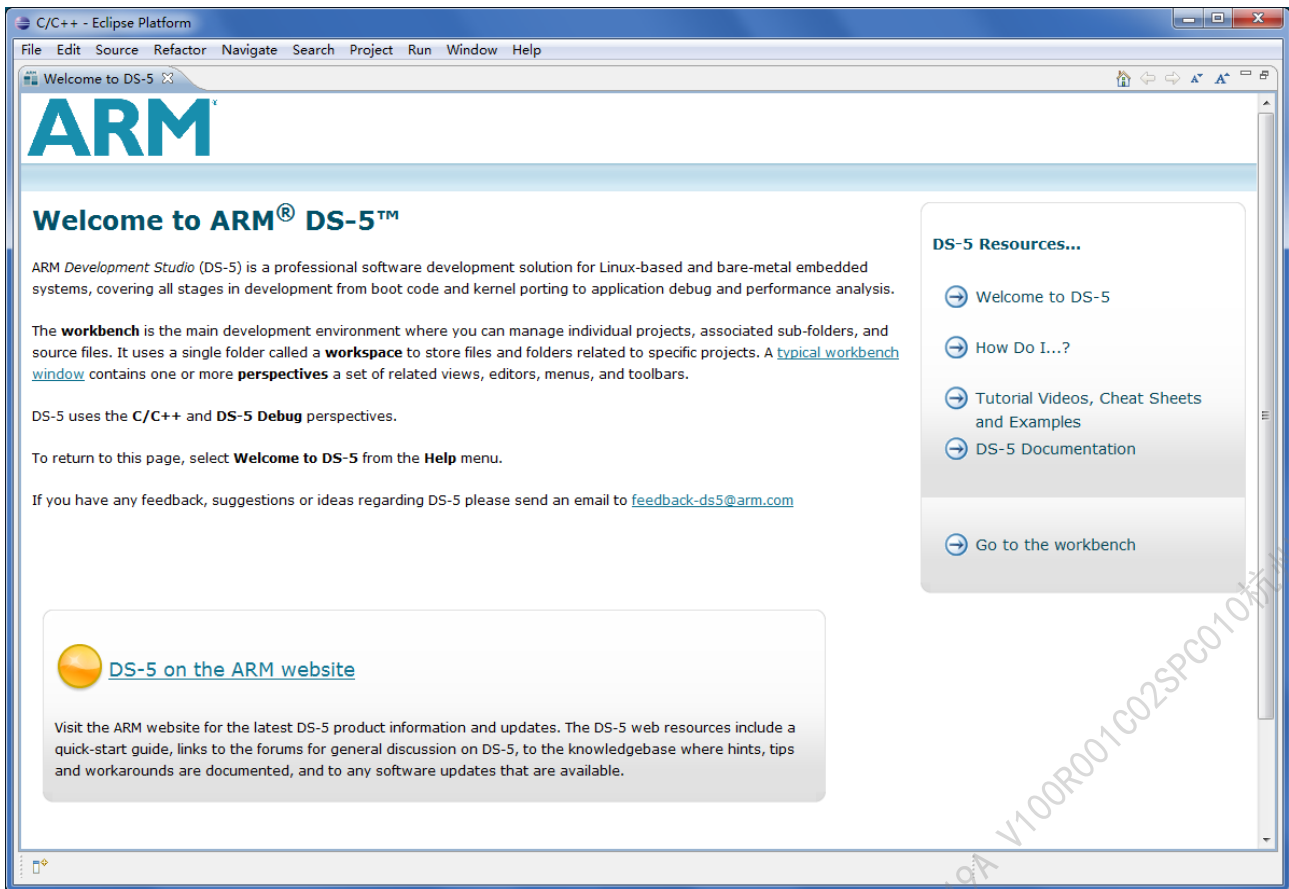
----结束

4.3.1 安装 ARM Development Studio 5

ARM Development Studio 5 是由 ARM 公司提供的 DS-5 Eclipse 安装程序。安装前，请先阅读 ARM 的相关文档。安装完成后启动 DS-5 Eclipse，如图 4-1 所示。



图4-1 DS-5 Eclipse 启动界面



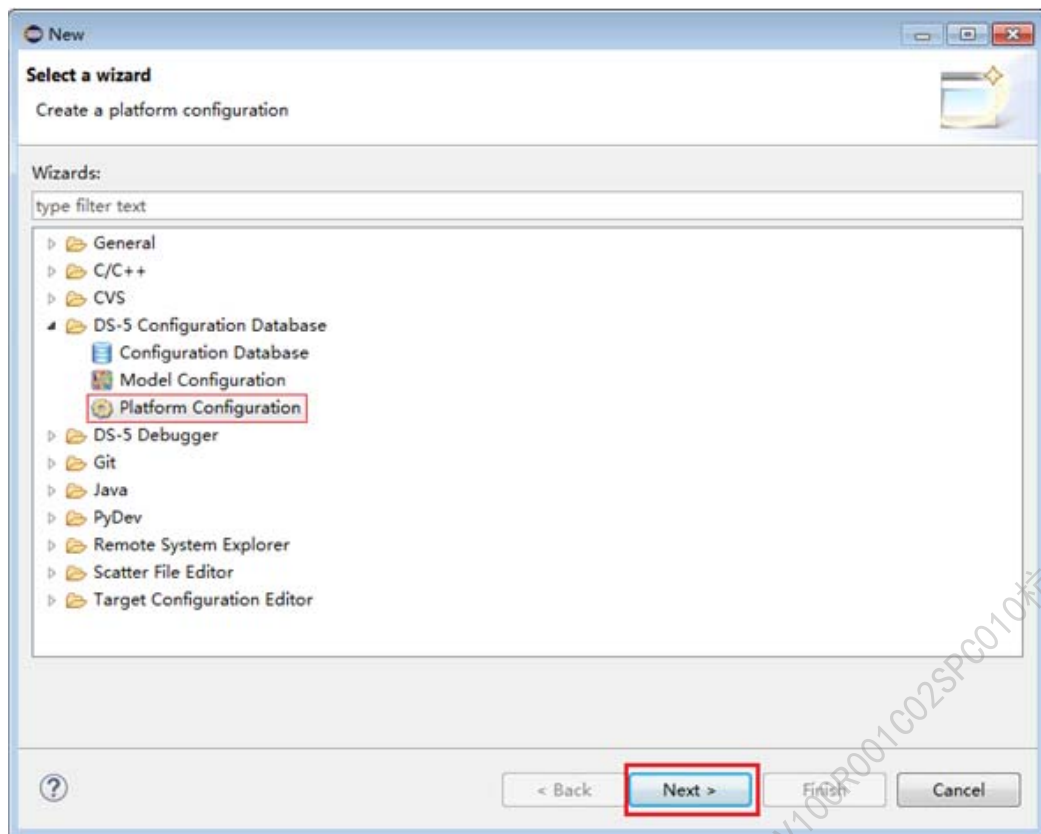
4.3.2 新建目标平台配置数据库

新建目标平台配置数据库的步骤如下：

- 步骤 1. 选择【File】→【New】→【Other】，在弹出的对话框中选择 DS-5 Configuration Database 文件夹下的 Platform Configuration, 然后点击【Next >】按钮，按照提示向下进行配置。



图4-2 平台配置界面



步骤 2. 连接仿真器，【菜单】—【ARM DS-5 v5.24.1】—【Debug Hardware】—【Debug Hardware Config IP(5.24.1)】，进入软件界面点击【scan】按钮进行扫描，扫描出仿真器后配置仿真器 IP 与当前 PC 机处于同一网段。

图4-3 Config IP 界面

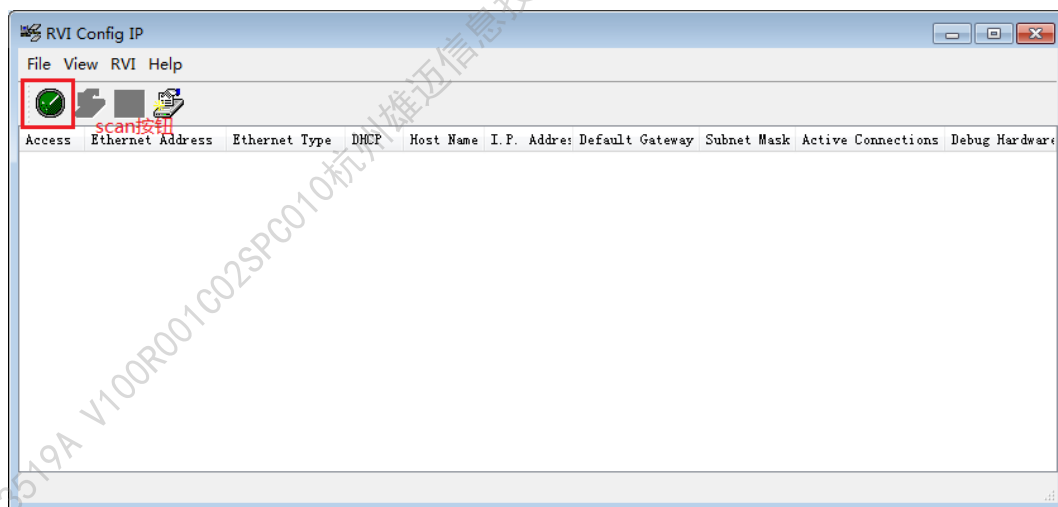




图4-4 Config IP 扫描界面

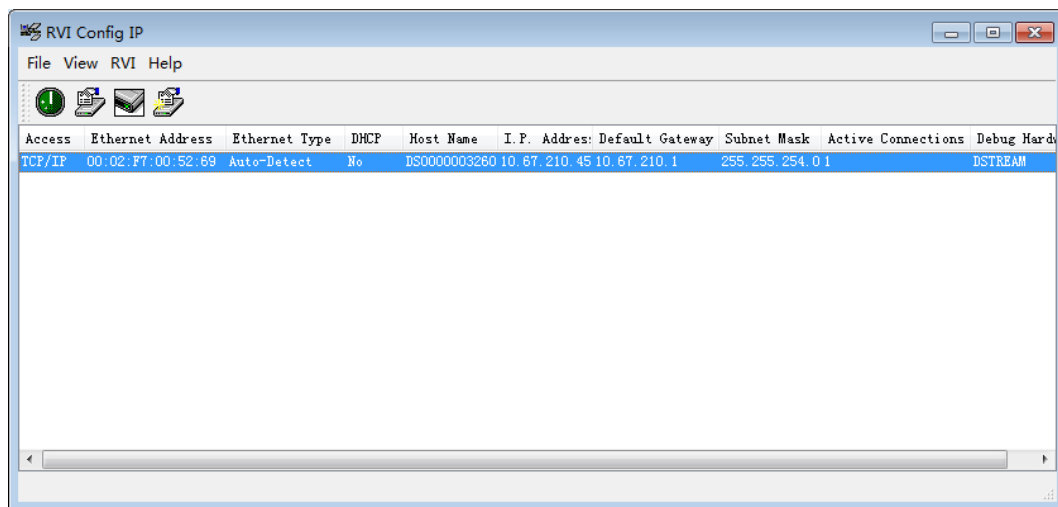
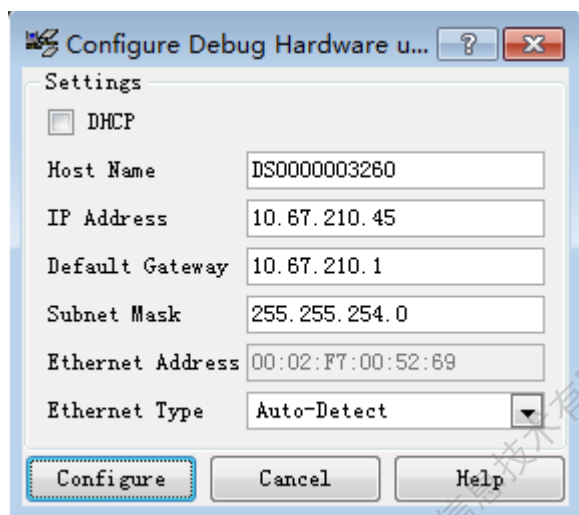


图4-5 配置仿真器 IP 界面



- 步骤 3. 返回 DS-5 Eclipse 界面，选择【Automatic/simple platform detection(Recommended)】点击下一步，系统会自动进行扫描，将仿真器 IP 地址输入【Connection Address:】，点击【Next >】，勾选上 Debug target after saving configuration，点击【Next >】，点击【Create New Database】，输入名字，点击【OK】，点击【Next >】，修改【Platform Manufacturer】内容为“Hisilicon”，修改【Platform Name】内容为“Hi35XX”，点击【Finish】完成平台数据库配置。



图4-6 配置平台数据库界面—Create Platform Configuration

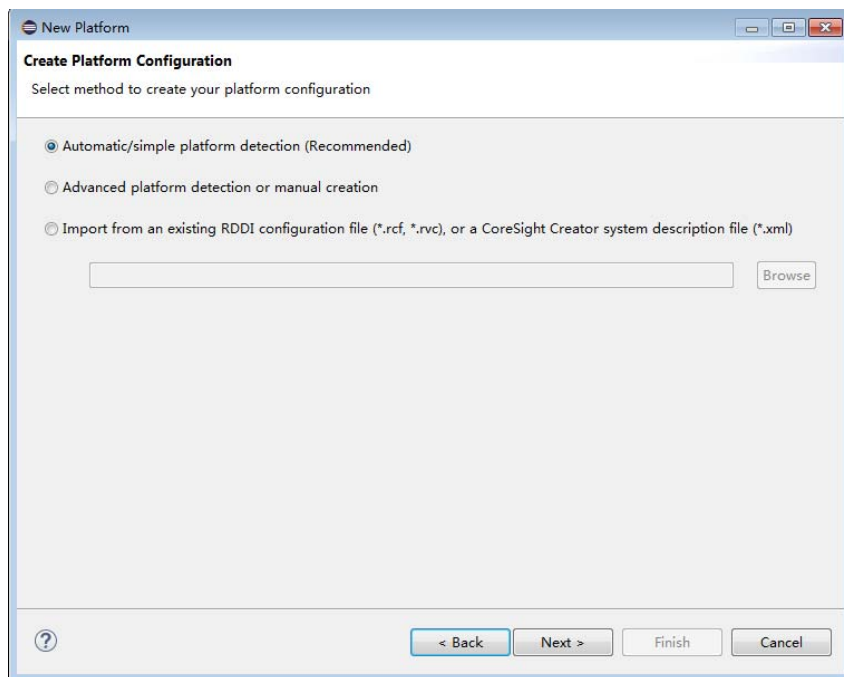


图4-7 配置平台数据库界面—Debug Adapter Connection

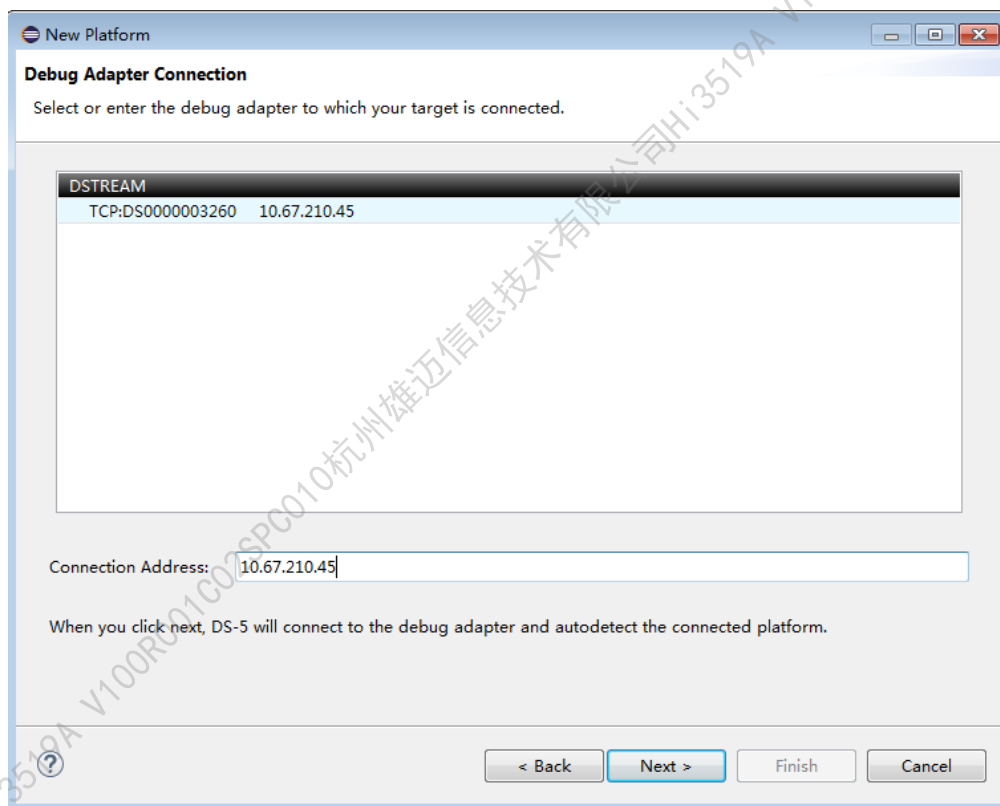




图4-8 配置平台数据库界面—Summary

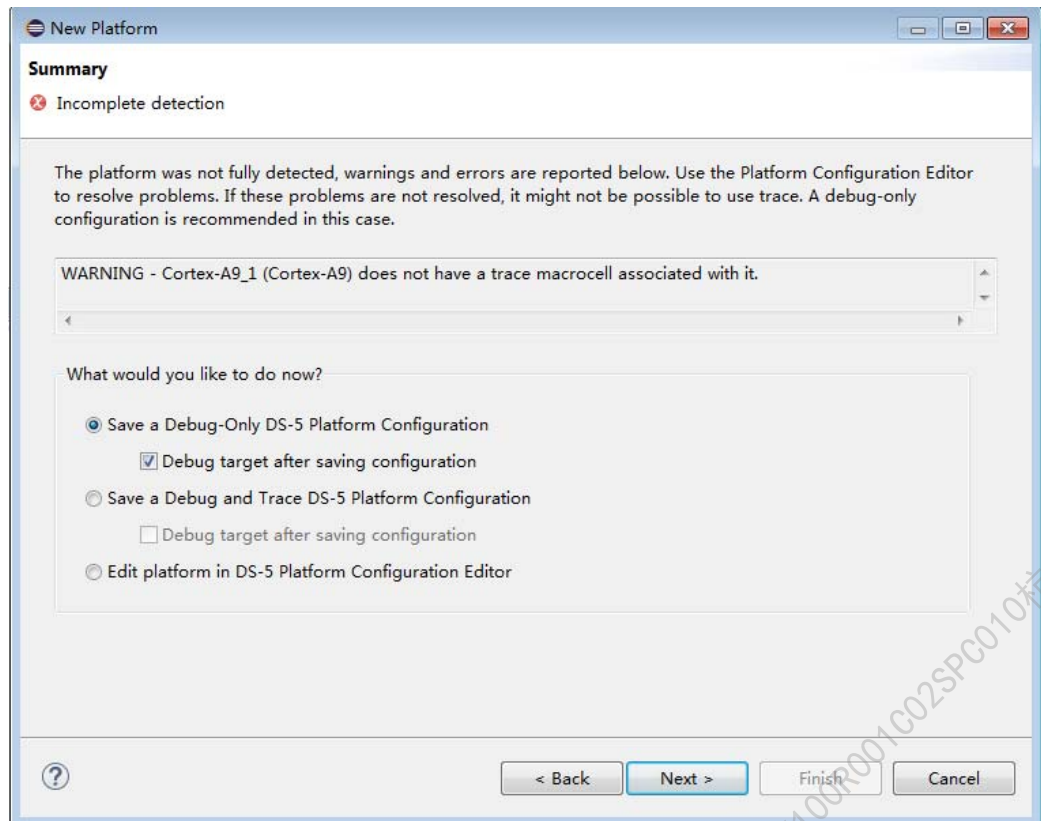




图4-9 配置平台数据库界面—DS-5 Configuration Database—Create New Database

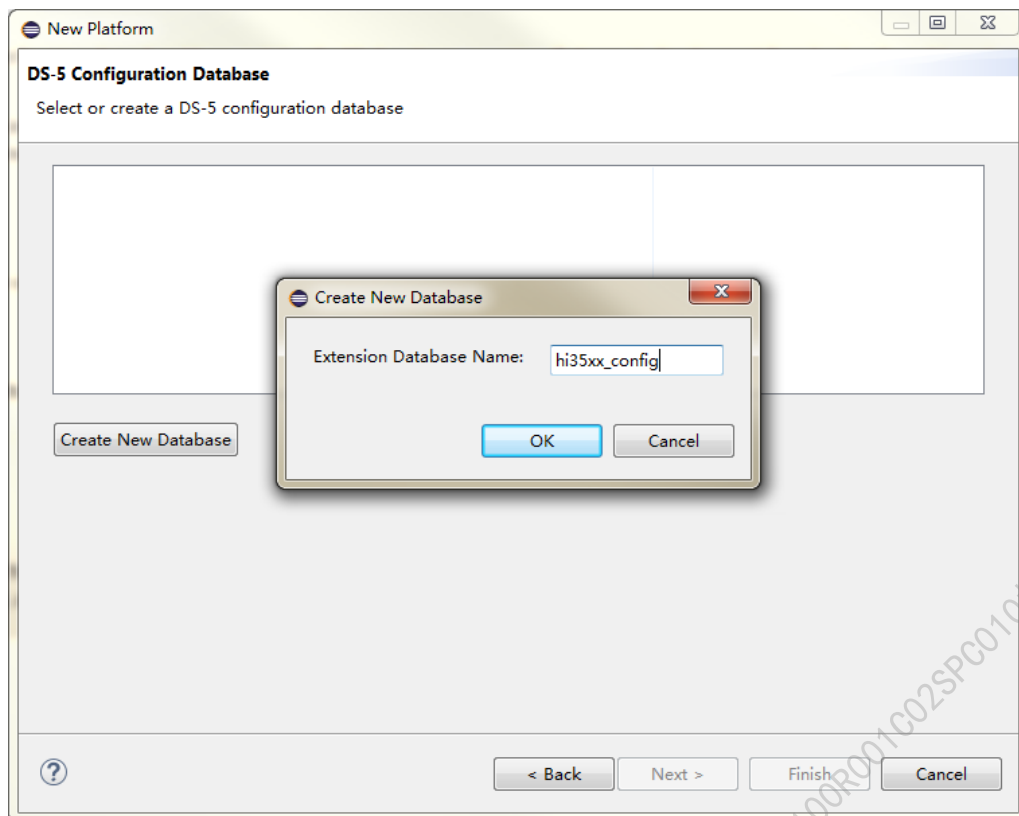




图4-10 配置平台数据库界面—DS-5 Configuration Database—完成 “Create New Database”

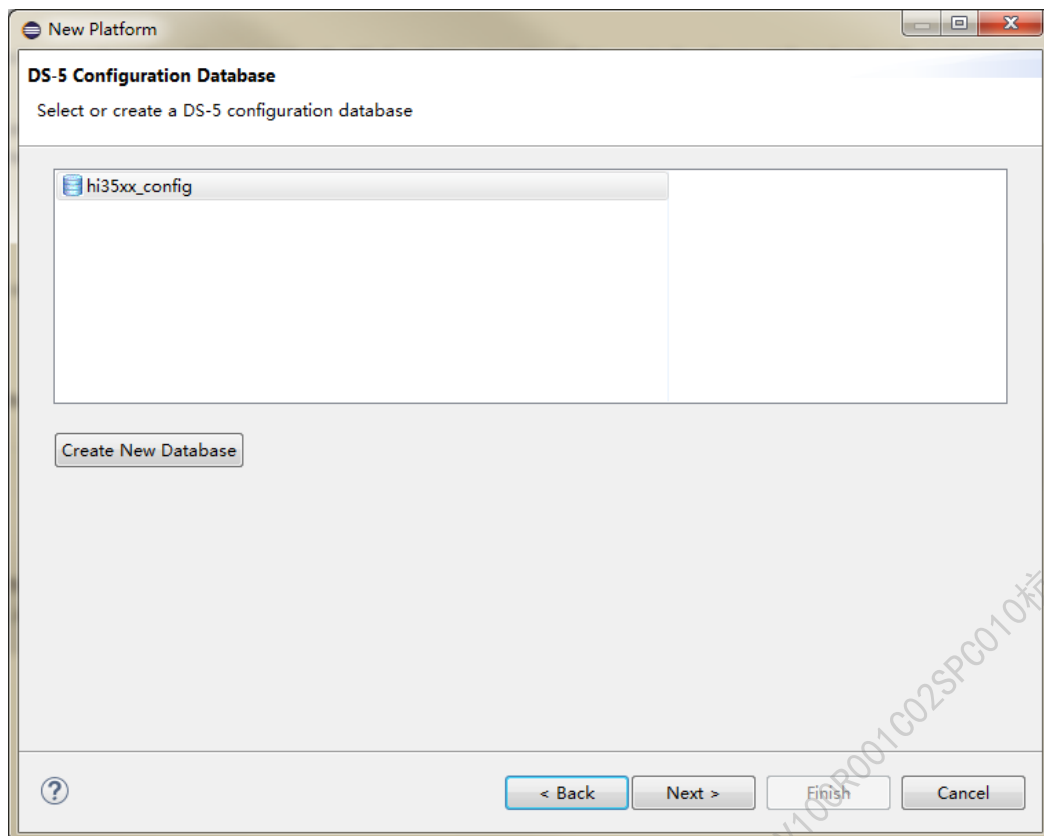




图4-11 配置平台数据库界面—Platform Information

New Platform

Platform Information

Use this page to enter identification details for the platform.

Platform Manufacturer: Imported

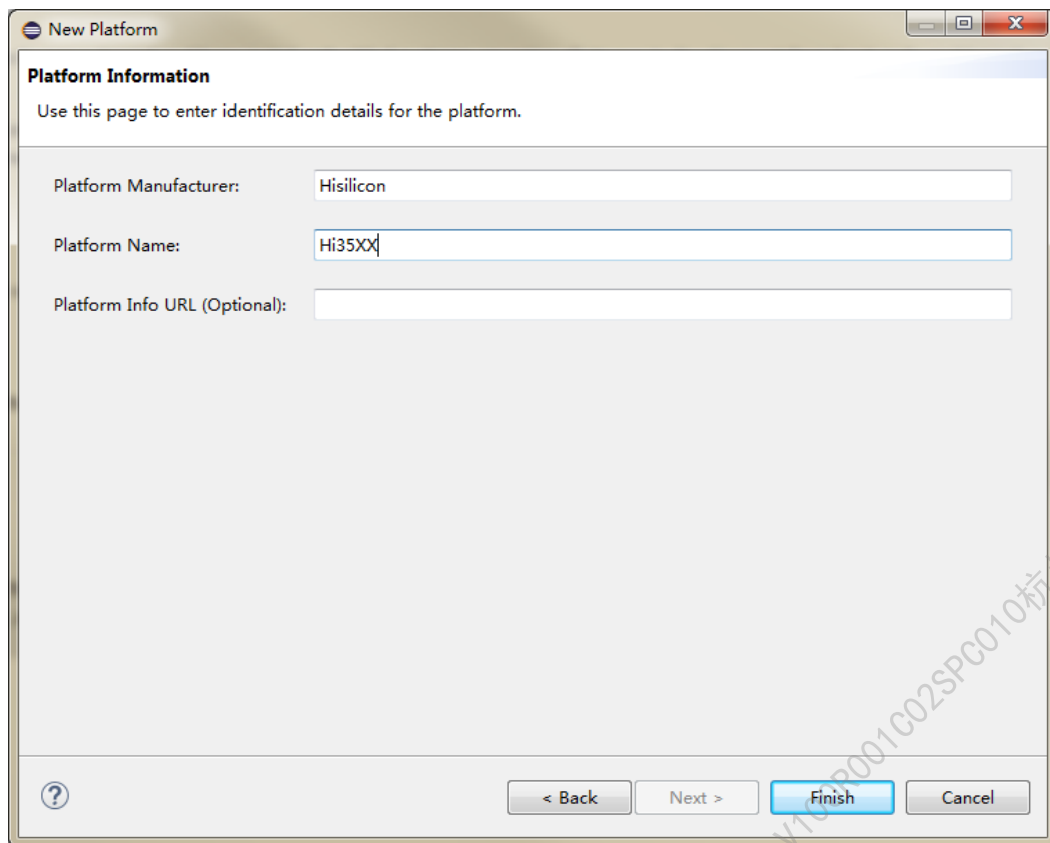
Platform Name: Imported Platform

Platform Info URL (Optional):

? < Back Next > Finish Cancel



图4-12 配置平台数据库界面—完成“Platform Information”配置



----结束

4.3.3 连接目标平台

连接到目标平台上的具体步骤如下：

- 步骤 1. 上一步点击【Finish】后会出现一个会话窗口。在名字域内，找到 DS-5 调试器，右击—【New】，选择刚才创建的 Hisilicon-Hi35XX。
- 步骤 2. 在【Connection】标签页选择新添加的目标平台配置数据库：【Hisilicon】→【Hi35XX】→【Bare Metal Debug】→【Cortex-A53】，在文本框输入 DS-5 设备的 IP 地址，如图 4-14。
- 步骤 3. 在【Debugger】标签页选中【Connect Only】选项，如图 4-15 所示。
- 步骤 4. 单击【Debug】按钮连接目标平台。

----结束



图4-13 Debug Configurations 窗口

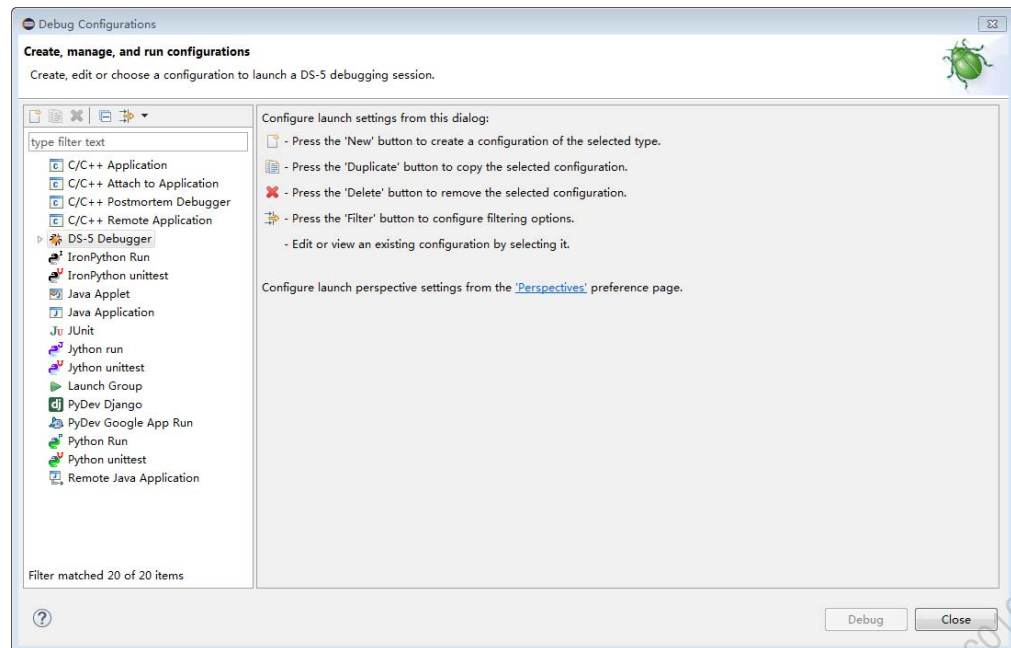




图4-14 Debug Configurations 窗口—选择新添加的目标平台配置数据库并输入 DS-5 设备的 IP 地址

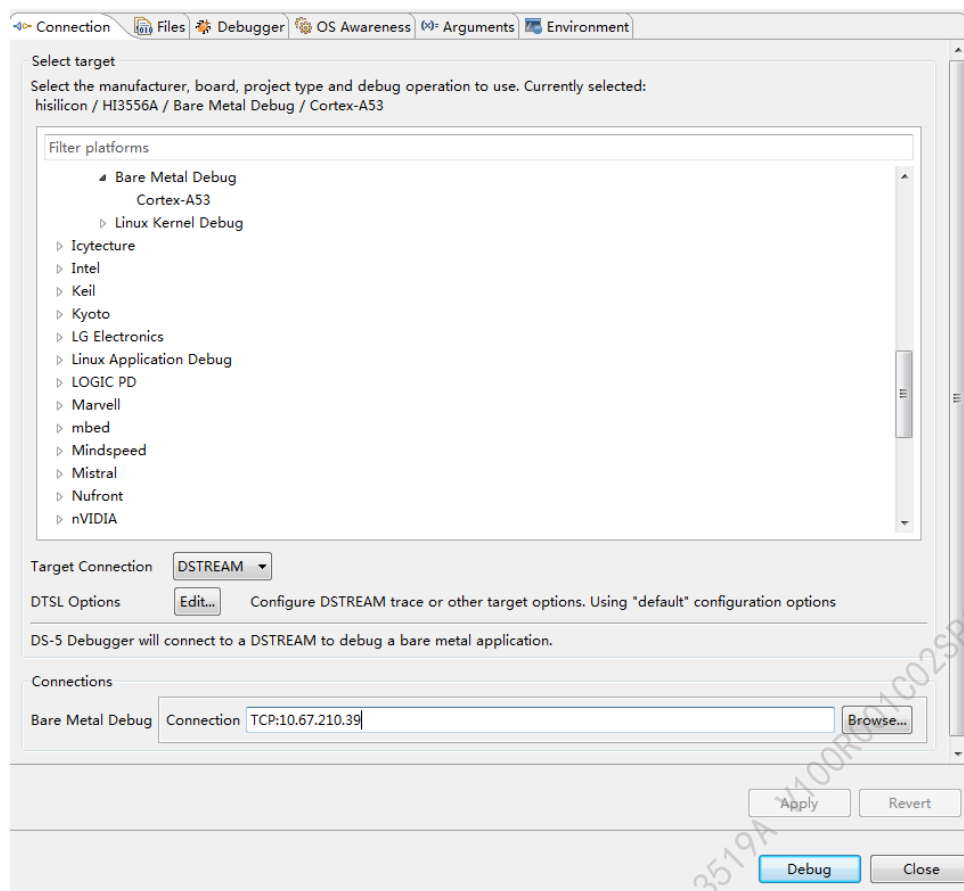




图4-15 Debug Configurations 窗口—勾选 “Connect only”

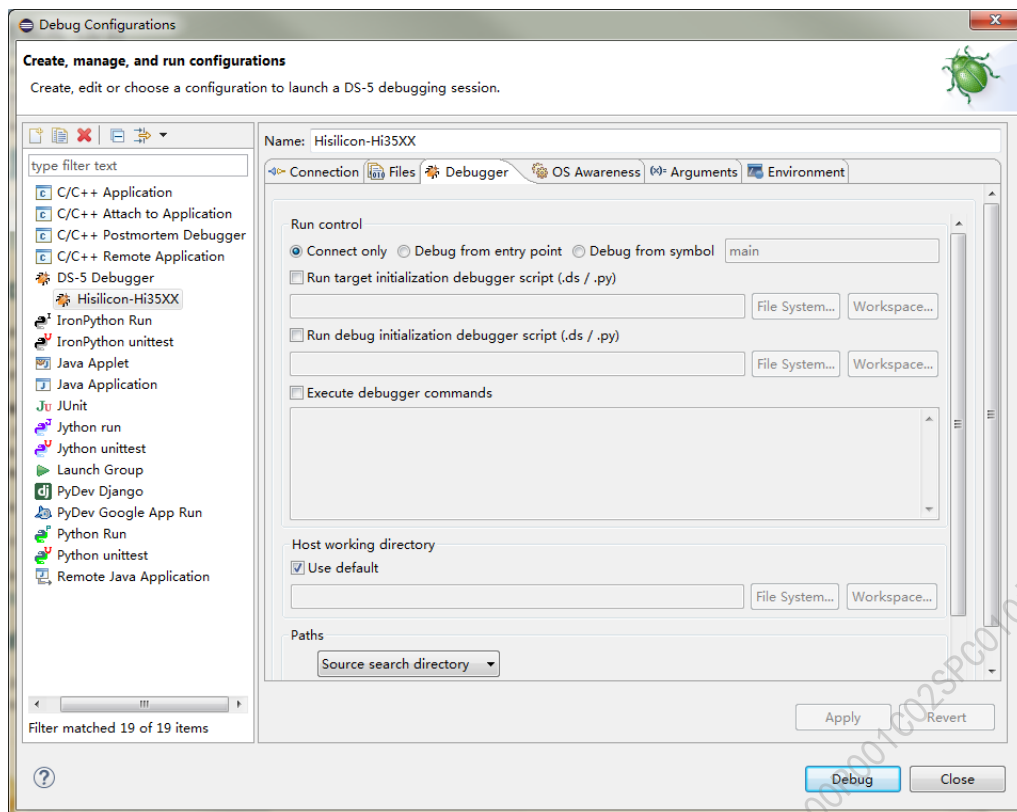
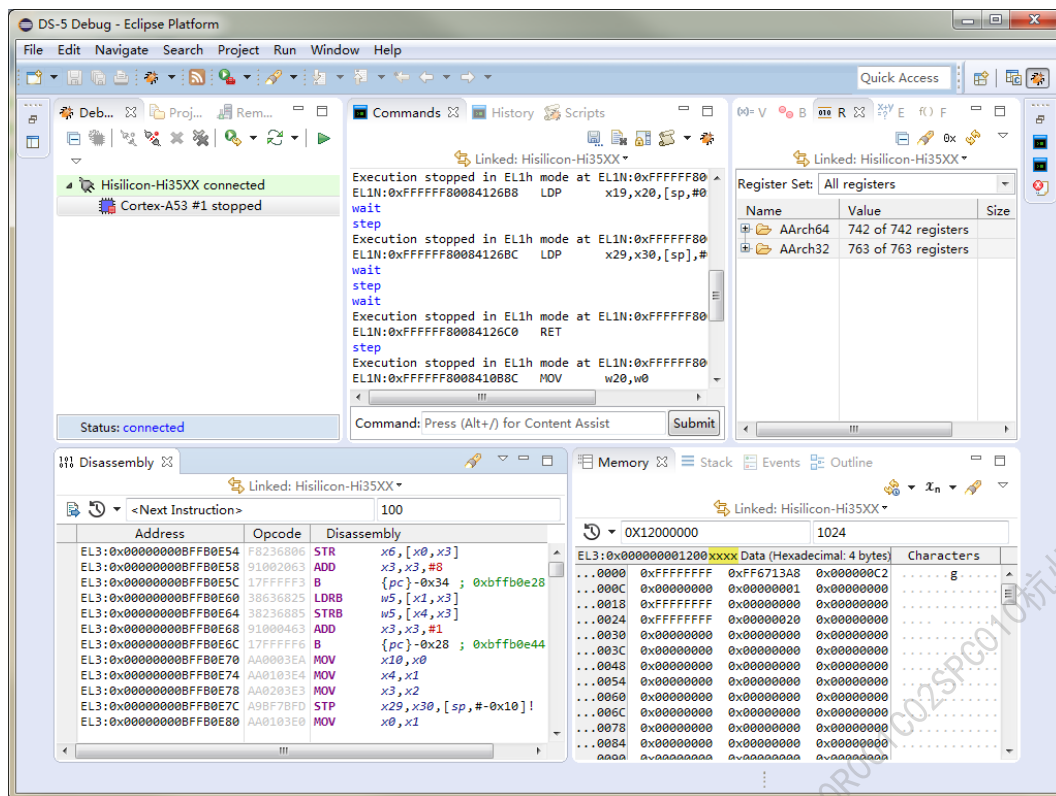







图4-16 DS-5 Debug – Eclipse Platform 窗口



4.4 使用仿真器烧写 Flash

4.4.1 内存初始化

在【Scripts】窗口单击图标  导入内存初始化脚本，单击图标  运行内存初始化脚本（如果此时仿真器处于运行状态，则需在【Debug Control】窗口单击按钮  暂停仿真器）。如图 4-17 所示。

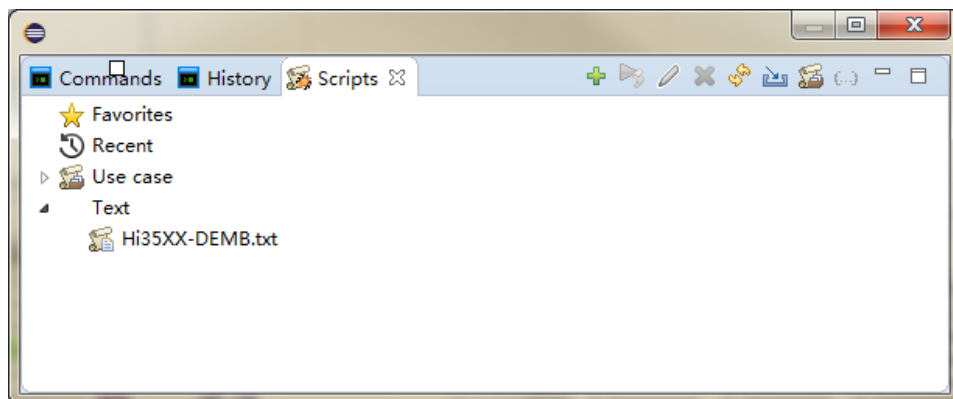


注意

内存初始化脚本为目录 osdrv/tools/pc/uboot_tools 下的 .ds、.py 或 .txt 格式文件。



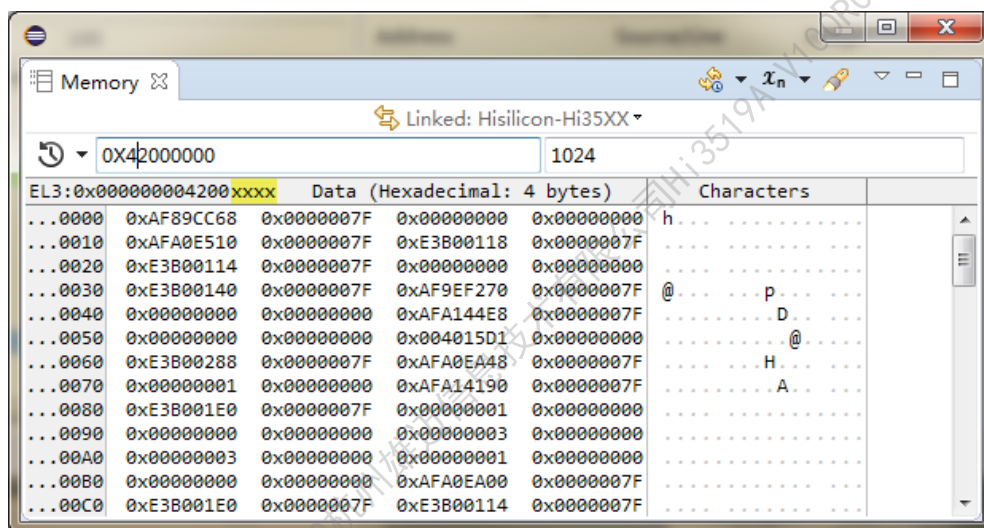
图4-17 脚本窗口



可通过以下方式验证内存初始化成功与否：


在【Memory】窗口输入内存地址（如 0x42000000），回车后查看表格是否显示当前内存区域的值。如果表格中显示数值，且能够成功改写则代表内存初始化成功。改写内存值的方法为：双击某个表格框（如 0x42000000 位置），输入新值（如 0x12345678）后回车，观察此框中值是否变成新值，如图 4-18 所示。

图4-18 Memory 窗口




4.4.2 下载 U-Boot 映像

步骤如下：

- 步骤 1. 在【Memory】窗口的单击按钮  弹出如图 4-19 所示菜单。
- 步骤 2. 选择【Import Memory】选项弹出映像下载窗口，下载 u-boot 映像到内存地址（如 0x42000000），如图 4-20。
- 步骤 3. 在【Registers】窗口修改 PC 指针值为 0x42000000，如图 4-21 所示。



步骤 4. 单击【Debug Control】窗口按钮  启动 U-Boot，此时可通过串口查看 U-Boot 启动信息。

----结束

图4-19 Memory 下拉窗口

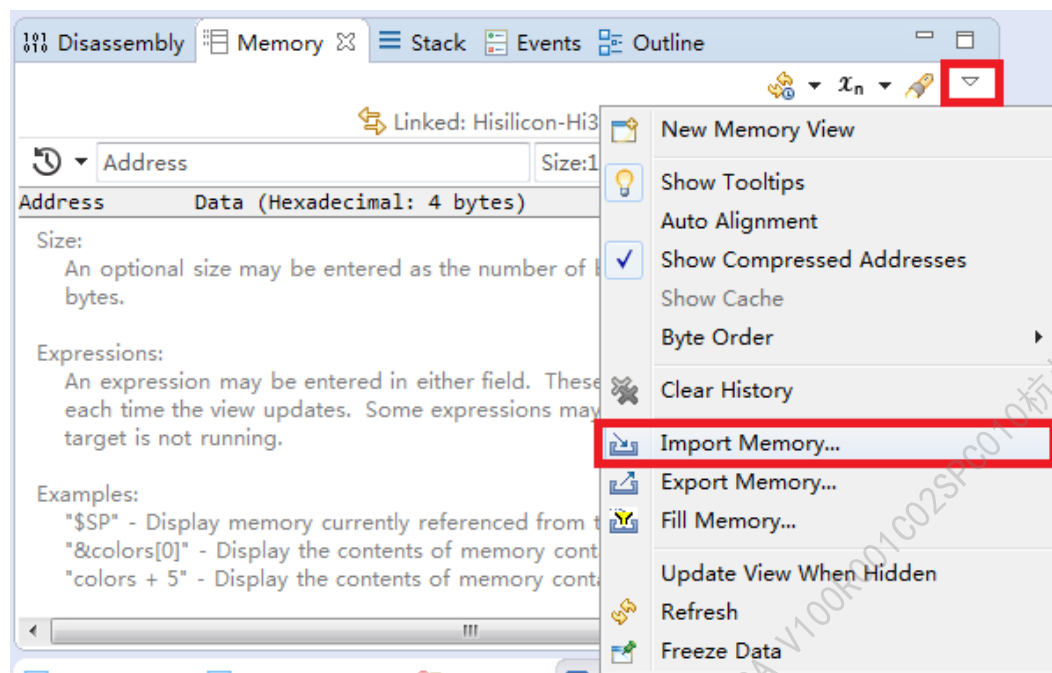




图4-20 Memory Importer 窗口

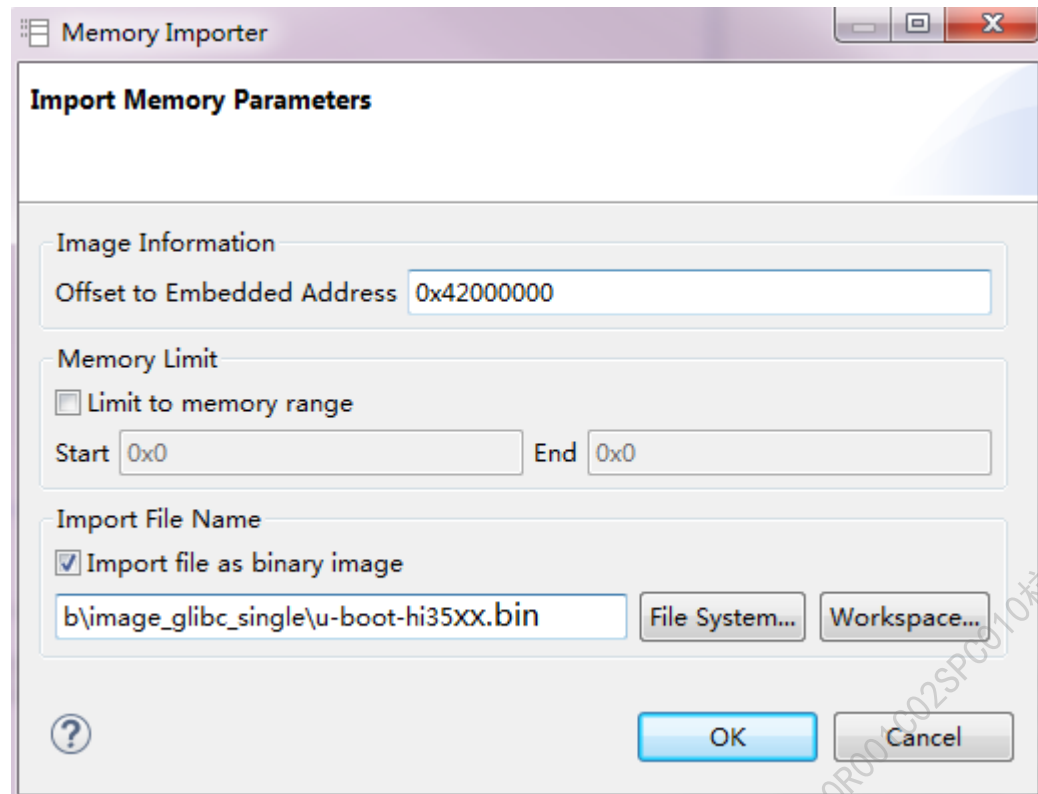
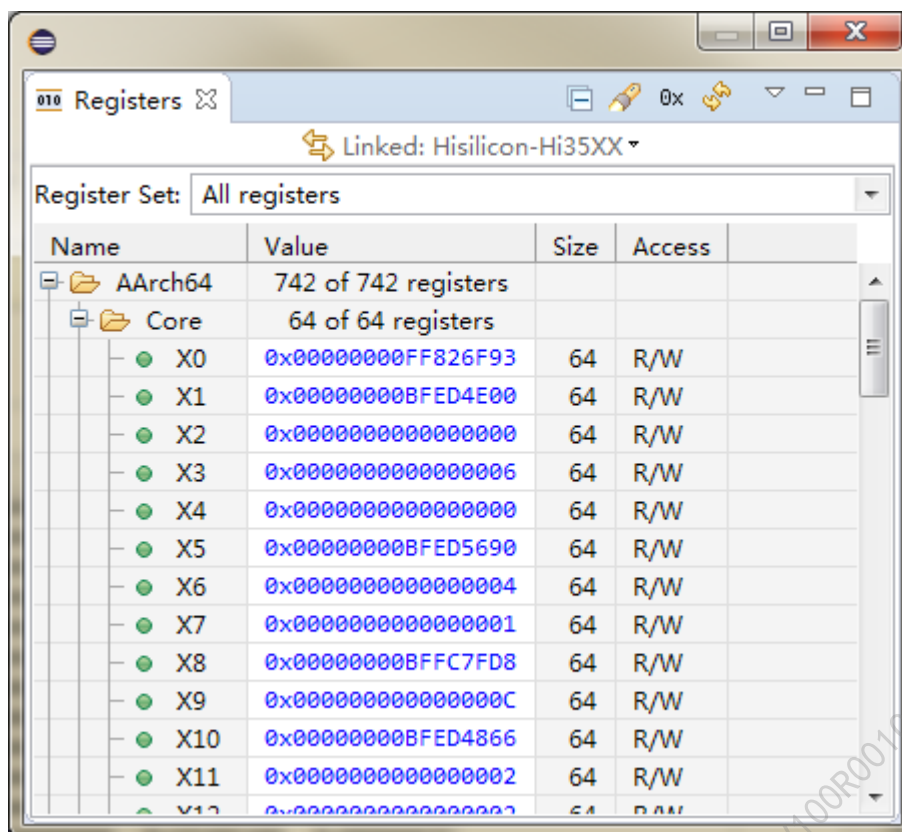




图4-21 Registers 窗口



4.4.3 烧写映像

U-Boot 启动后，通过串口将内存中的 U-Boot 映像写入启动介质中。

以 SPI-Nor Flash 为例，其烧写步骤如下：

```
# sf probe 0 /*探测并初始化SPI-Nor flash*/
# sf erase 0 0x100000 /*擦除 1M大小*/
# sf write <ddr_addr> 0 0x100000 /*从内存写入SPI-Nor Flash*/
# reset /*重启单板*/
```



说明

Hi3519AV100 平台的<ddr_addr>可用地址 0x22000000。



5 附录

5.1 u-boot 命令说明

5.1.1 SPI-Nor 块保护命令

常用的 SPI Nor Flash 上都提供了块保护位(Block Protect: 以下简称 BP)来保护数据安全。

通过设置状态寄存器 (Status Register: 以下简称 SR) 中的 BP0、BP1、BP2、BP3 (某些厂家的芯片没有 BP3 或者存在 BP4) 几个 Bit 为 1(使能状态), 使器件中某些对应的块进入写保护状态, 这些 BP 位为非易失性位 (Non-volatile Bit), 设置之后可以掉电保持之前状态。

有的厂商还提供了对块保护锁定方向的设置, 可以设置保护的块从器件顶部 Top 开始还是从器件底部 Bottom 开始。通过设置配置寄存器 (Config Register: 以下简称 CR) 中的 TBPROT 位 (某些厂家的芯片 TBPROT 位位于 SR) 设置写保护锁定起始地址是从低地址 (Bottom) 还是从高地址 (Top) 开始。通常这个设置位属于 OTP (One Time Program) 类型, 默认状态为 0: BP 开始于 Top 部 (高地址), 一旦状态被设置为 1: BP 开始于 Bottom (低地址), 且将无法更改。

根据实际应用, 我们控制器从初始化开始就将 TBPROT 位置为 1, 即从 Bottom 低地址开始保护。

SPI-Nor 器件状态寄存器 SR 的默认初始值中, 所有 BP 位都为 0 (去使能状态), 此时器件上所有的块都处于未保护状态, 可以任意进行擦写操作。

设置所有 BP 位都为 1 (使能状态), 将使器件上所有的块都处于写保护状态, 任何擦写操作都将无效。

块保护以块 (Block) 为基本单位, 根据 BP 位的使能状态, 转换为十进制 level 值来设置锁定的块保护的 range。对于 3 个 BP 位的器件, 在 BP[0:0:0]到 BP[1:1:1]之间, level 的取值范围为 0~7; 对于 4 个 BP 位的器件, 在 BP[0:0:0:0]到 BP[1:1:1:1]之间, level 的取值范围 0~10 (或者 0~9, 这是因为最小的锁定区域不能小于 1 Block)。

块保护锁定区域根据不同厂家不同器件而有所差异, 如表 5-1 所示:



表5-1 不同厂家不同器件的块保护锁定区域与 BP Level 对应表

等级	MXIC MX25L-					ESMT F25L-
	12835F	25635F	25735F	6406E	1606E	64QA
0	解锁	解锁	解锁	解锁	解锁	解锁
1	0-64KB	0-64KB	0-64KB	0-4MB(64Block)	全锁 2MB(32Block)	0-4MB(64Block)
2	0-128KB	0-128KB	0-128KB	0-6MB(96Block)	0-1MB(16Block)	0-6MB(96Block)
3	0-256KB	0-256KB	0-256KB	0-7MB(112Block)	0-1.5MB(24Block)	0-7MB(112Block)
4	0-512KB	0-512KB	0-512KB	0- 7.5MB(120Block)	0- 1.75MB(28Block)	0-7.5MB(120Block)
5	0-1MB	0-1MB	0-1MB	0- 7.75MB(124Block)	0- 1.875MB(30Block)	0-7.75MB(124Block)
6	0-2MB	0-2MB	0-2MB	0- 7.875MB(126Block)	0- 1.9375MB(31Block)	0- 7.875MB(126Block)
7	0-4MB	0-4MB	0-4MB	全锁 8MB (128Block)	全锁 2MB(32Block)	全锁 8MB (128Block)
8	0-8MB	0-8MB	0-8MB	-	-	-
9	全锁 16MB	0-16MB	0-16MB	-	-	-
10	-	全锁 32M	全锁 32M	-	-	-
等级	SPANSION S25FL-		WINBOND W25Q-			
	127S	256S	128FV	128BV	256FV	64FV
0	解锁	解锁	解锁	解锁	解锁	解锁
1	0-256KB	0-512KB	0-256KB	0-256KB	0-64KB	0-128KB
2	0-512KB	0-1MB	0-512KB	0-512KB	0-128KB	0-256KB
3	0-1MB	0-2MB	0-1MB	0-1MB	0-256KB	0-512KB
4	0-2MB	0-4MB	0-2MB	0-2MB	0-512KB	0-1MB
5	0-4MB	0-8MB	0-4MB	0-4MB	0-1MB	0-2MB
6	0-8MB	0-16MB	0-8MB	0-8MB	0-2MB	0-4MB
7	全锁 16MB	全锁 32MB	全锁 16MB	全锁 16MB	0-4MB	全锁 8MB
8	-	-	-	-	0-8MB	-
9	-	-	-	-	0-16MB	-
10	-	-	-	-	全锁 32MB	-



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

根据 SPI-Nor Flash 上的块保护机制，u-boot 下新增 SPI-Nor 的块保护命令 lock。命令格式如下：

- sf lock

可以查看当前设置的 BP level 值和 level 的取值范围以及当前锁定的区域范围，同时打印命令说明信息。如图 5-1 所示。

图5-1 查看当前块保护信息

```
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

锁定所有的块（整个器件），在表 5-1 所示中，等同于设置等级 level 为最大值，如图 5-2 所示。



图5-2 锁定整个器件

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

- sf lock 0

解除当前块保护锁定状态，此时器件上所有的块都处于未保护状态，可以任意进行擦写操作。如图 5-3 所示。

图5-3 解除当前锁定状态

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

- sf lock <level>

设置 BP level 值，根据 level 值，按照表 5-1 所示来保护对应的区域，这样处于块保护区域的块不可以进行正常擦写，如图 5-4 所示。

图5-4 通过设置 level 值锁定指定区域

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