



Hi35xx U-boot 移植应用 开发指南

文档版本 00B05
发布日期 2016-08-30

版权所有 © 深圳市海思半导体有限公司 2016。保留一切权利。

非经本公司书面许可，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 Hi35xx 单板上如何移植和烧写 U-boot（Hi35xx 单板的 Bootloader）的相关操作及如何使用 ARM 调试工具。

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
Hi3519	V100
Hi3519	V101
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3516A	V100
Hi3516D	V100
Hi3559	V100
Hi3516C	V300

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
2016-08-30	00B05	第 5 次临时版本发布。 添加 Hi3516CV300 相关内容。
2016-07-25	00B04	第 4 次临时版本发布。 添加 Hi3559V100 相关内容。
2016-06-20	00B03	第 3 次临时版本发布。 添加 Hi3519V101 相关内容。
2016-05-13	00B02	第 2 次临时版本发布。 4.3.2 和 4.3.3 小节有修改。 添加 Hi3519V100 相关内容。
2015-12-10	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.....	2
2.1 U-boot 硬件环境	2
2.2 编译 U-boot	2
2.3 配置 DDR 存储器	3
2.4 配置管脚复用	3
2.5 生成最终使用的 U-boot 镜像.....	3
3 烧写 U-boot.....	4
3.1 概述.....	4
3.2 通过 bootrom 工具烧写 U-boot	4
3.3 Flash 的 U-boot 烧写方法.....	4
3.3.1 SPI Nor Flash 烧写方法	4
3.3.2 NAND Flash/SPI-Nand Flash 烧写方法	5
3.4 eMMC 的 U-boot 烧写方法	5
4 如何使用 ARM 调试工具	7
4.1 概述.....	7
4.2 ARM 调试工具简介.....	7
4.2.1 DS-5 Eclipse	7
4.2.2 DS-5 Debug	7
4.3 使用 ARM 调试工具.....	8
4.3.1 安装 ARM Development Studio 5	8
4.3.2 新建目标平台配置数据库	9
4.3.3 连接目标平台	14
4.4 使用仿真器烧写启动介质	17
4.4.1 内存初始化.....	17
4.4.2 下载 U-Boot 映像.....	18



4.4.3 烧写映像	20
5 附录.....	22
5.1 u-boot 命令说明	22
5.1.1 SPI NOR 块保护命令	22



插图目录

图 4-1 DS-5 Eclipse 启动界面	9
图 4-2 设备扫描窗口	10
图 4-3 芯片配置窗口	11
图 4-4 DS-5 命令提示符窗口	12
图 4-5 Preferences 窗口	13
图 4-6 Add configure Database 对话框	13
图 4-7 DS-5 Debug 窗口	14
图 4-8 Debug Configure 窗口	15
图 4-9 Debug Configure 窗口	16
图 4-10 Debug Configure 窗口	17
图 4-11 脚本窗口	18
图 4-12 Memory 窗口	18
图 4-13 Memory 窗口	19
图 4-14 Memory Importer 窗口	20
图 4-15 Registers 窗口	20
图 5-1 查看当前块保护信息	24
图 5-2 锁定整个器件	25
图 5-3 解除当前锁定状态	25
图 5-4 通过设置 level 值锁定指定区域	25



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	23



1 概述

1.1 概述

Hi35xx 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/hi351xx	Hi351xx 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。



2 移植 U-boot

2.1 U-boot 硬件环境

Hi35xx DMEB 板上的外围芯片包括 DDR SDRAM、NAND Flash、SPI Flash 和 SPI-NAND Flash，其具体型号见《Hi35xx Compatibility Test Report》。



说明

其中 Hi3518EV200、Hi3518EV201、Hi3516CV200、Hi3516CV300 不支持 NAND Flash。

2.2 编译 U-boot

当所有以上移植步骤完成后，就可以编译 U-boot，操作如下：

步骤 1. 配置编译环境

```
make ARCH=arm CROSS_COMPILE=arm-hisiv500-linux- hi3516a_config
```



说明

配置编译环境步骤，配置命令根据平台芯片系列的不同而有所调整：

- Hi3518EV200、Hi3518EV201、Hi3516CV200 平台：

```
make ARCH=arm CROSS_COMPILE=arm-hisiv500-linux- hi3518ev200_config
```

- Hi3519V100/Hi3519V101 平台：

```
make ARCH=arm CROSS_COMPILE=arm-hisiv500-linux- hi3519_config
```

- Hi3516CV300 平台：

```
make ARCH=arm CROSS_COMPILE=arm-hisiv500-linux- hi3516cv300_config
```

- Hi3559V100 平台：

```
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- hi3559_config
```

其中表述为 arm-hisivXXX-linux-表示两种情况，下文不再复述：

arm-hisiv500-linux- 对应 uclibc 工具链

arm-hisiv600-linux- 对应 glibc 工具链

步骤 2. 编译 U-boot



```
make ARCH=arm CROSS_COMPILE=arm-hisiv500-linux-  
编译成功后，将在U-boot目录下生成u-boot.bin。
```



说明

Hi3559V100 平台：

```
make ARCH=arm CROSS_COMPILE=arm-hisiv600-linux-
```

----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/uboot_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，对配置工作表中的标签页【mddrc_dmc1】、【mddrc_dmc2】和【mddrc_phy】进行修改。



说明

Hi3559V100、Hi3519V100/Hi3519V101 平台需要配置标签页【ddrc0_init】。

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标签页【muxctrl_reg】进行修改。

2.5 生成最终使用的 U-boot 镜像

完成配置表格的修改后，保存表格。单击表格第一个标签页上的按钮【Generage reg bin file】，注意只能点此按钮，不能点其他的。生成临时文件 reg_info. bin。

将临时文件 reg_info. bin 和编译 u-boot 得到的 u-boot.bin 都拷贝到“osdrv/tools/pc/uboot_tools/”目录下，执行命令：

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

其中 **u-boot-final.bin** 就是能够在单板上运行的 U-boot 镜像。



说明

其他平台除生成的临时文件名有差异外，步骤上与 Hi3516A 平台一致。



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 fastboot 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 Hi35xx 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：Hi3516A/Hi3516D 请参考《Fastboot 工具使用说明》，Hi3559V100、Hi3519V100/Hi3519V101、Hi3518EV20X、Hi3516CV300 请参考《HiTool 工具平台使用指南》和《HiBurn 工具使用指南》。

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：

```
hisilicon# mw.b 0x82000000 ff 0x100000 /* 对内存初始化*/
hisilicon# tftp 0x82000000 u-boot-final.bin /*U-boot下载到内存*/
hisilicon# sf probe 0 /*探测并初始化SPI Nor flash*/
hisilicon# sf erase 0x0 0x100000 /*擦除 1M大小*/
hisilicon# sf write 0x82000000 0x0 0x100000 /*从内存写入SPI Nor Flash*/
```

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



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：

```
hisilicon# nand erase 0 100000          /*擦除 1M大小*/  
hisilicon# mw.b 0x82000000 ff 100000    /* 对内存初始化*/  
hisilicon# tftp 0x82000000 u-boot-final.bin /*U-boot下载到内存*/  
hisilicon# nand write 0x82000000 0 100000 /*从内存写入NAND Flash*/
```

步骤 2. 重启系统可以看到 U-boot 烧写成功。

----结束



说明
Hi3518EV200/Hi3518EV201/Hi3516CV200/Hi3516CV300 不支持 NAND Flash。



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：

```
hisilicon# mw.b 0x82000000 0xff 0x80000 /* 对内存初始化*/  
hisilicon# tftp 0x82000000 u-boot-final.bin /* U-boot下载到内存*/  
hisilicon# mmc write 0 0x82000000 0 0x400 /*从内存写入eMMC*/
```



说明
注意：Hi3516A/Hi3516D 不支持 eMMC 烧写。

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

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，可以很轻松地为用户提供 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

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。首先运行 Debug Hardware Configure 生成芯片配置文件，然后使用该配置文件生成目标平台配置数据库，再将目标平台配置数据库添加到系统中。
- 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。

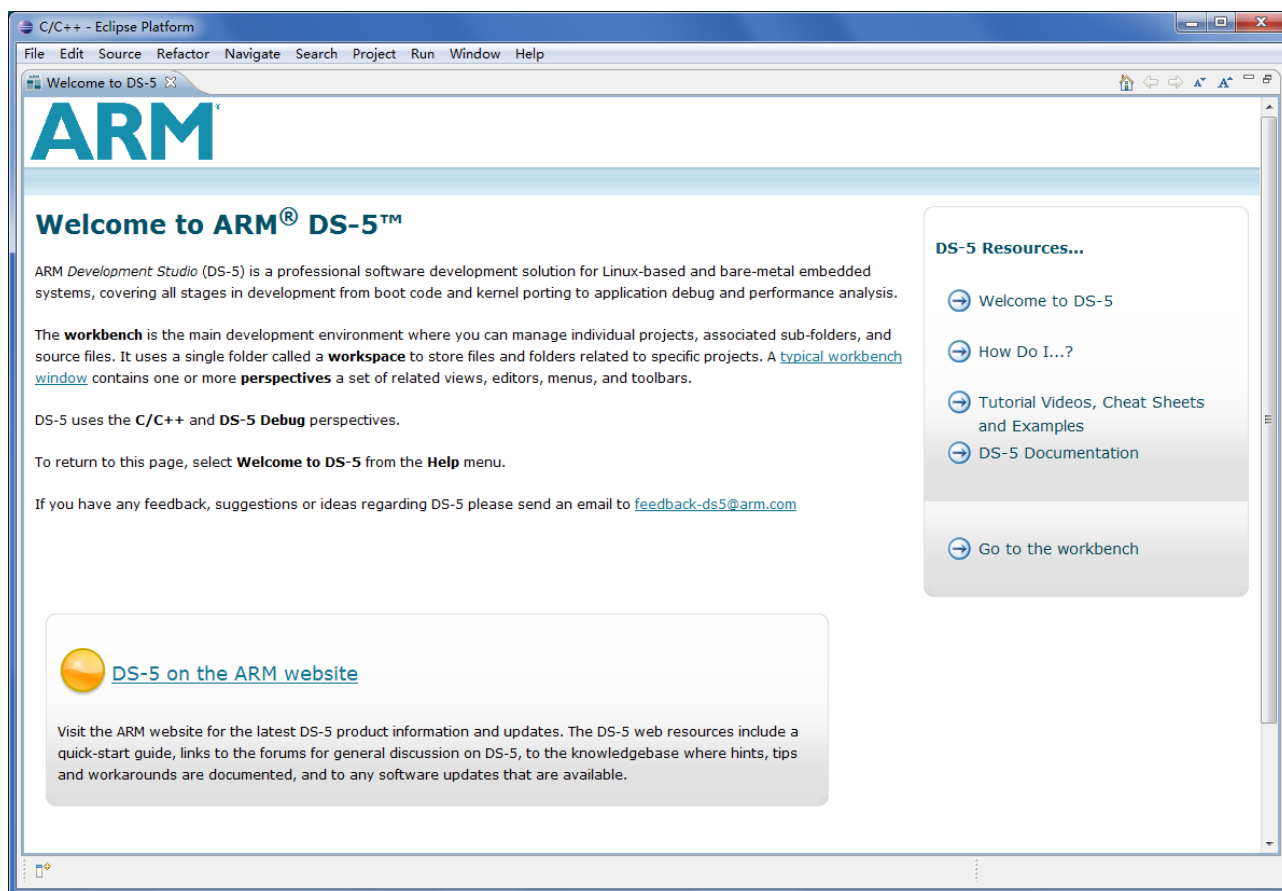
----结束

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 Windows 【Start】→【All Programs】→【ARM DS-5】→【Debug Hardware】→【Debug Hardware Configure】，运行 Debug Hardware Configure 程序扫描连接中的仿真器，选中指定的仿真器后点击【Connect】按钮，如图 4-2。在弹出的窗口中按红框中所示进行配置，点击【Auto Configure】按钮自动生成芯片配置，如图 4-3 所示。退出并保存该配置文件到指定路径，如 D:\DS-5\hi3516a.rvc。
- 步骤 2. 生成目标平台配置数据库。运行 DS-5 Command Prompt 程序，执行程序 cdbimporter.exe 读取配置文件，生成目标平台配置数据库，如图 4-4 所示。其操作步骤如下。
 - 运行 cdbimporter.exe，读取芯片配置文件 hi3516a.rvc。
 - 指定可识别目标平台的源数据库路径，直接回车，此处保持默认。
 - 自定义保存目标平台数据库的路径，如 D:\DS-5\database_hi3516a。
 - 自定义平台制造商，如“Hisilicon”。
 - 自定义平台名称，如“Hi3516A”。



步骤 3. 添加目标平台配置数据库到系统。从主菜单选择【Window】→【Preferences】，打开 Preferences 窗口，在配置树中选择【DS-5】→【Configuration Database】，如图 4-5。点击【add】按钮，在弹出的对话框中指定路径为保存目标平台配置数据库的路径 D:\DS-5\database_hi3516a，如图 4-6。

----结束

图4-2 设备扫描窗口

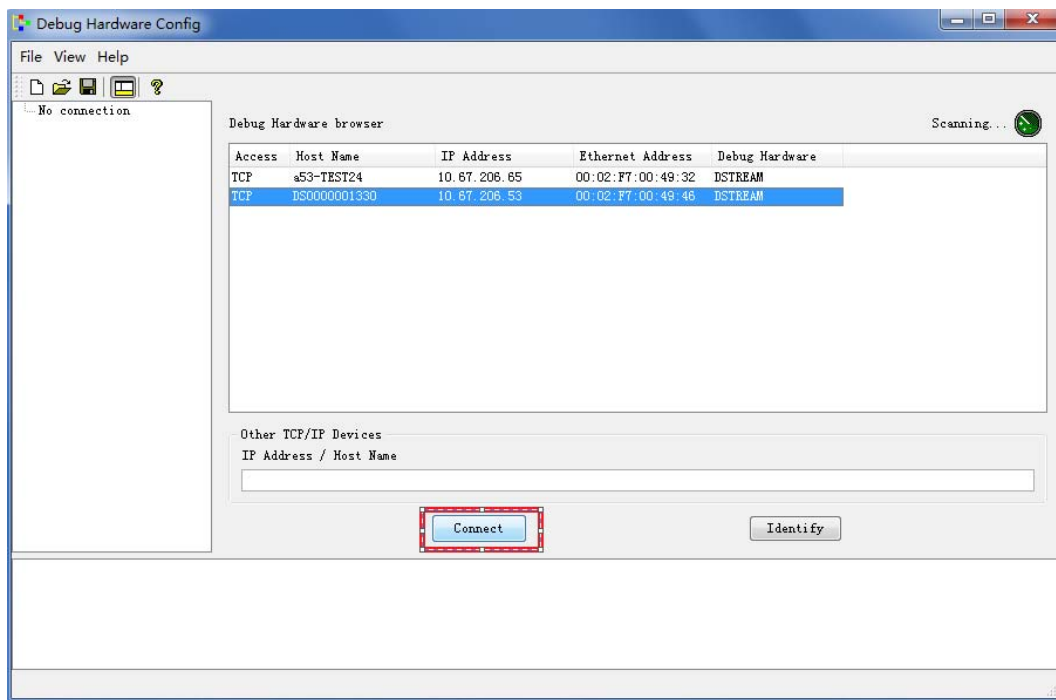




图4-3 芯片配置窗口

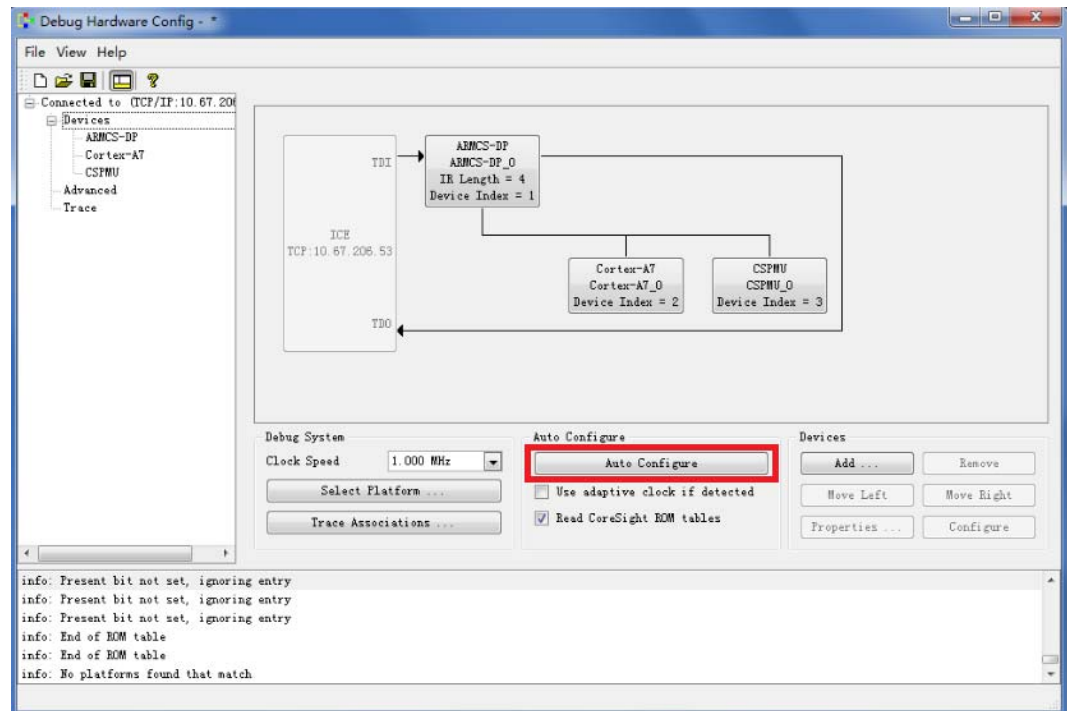




图4-4 DS-5 命令提示符窗口

```
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>cdhimporter.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\28241412\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.
```



图4-5 Preferences 窗口

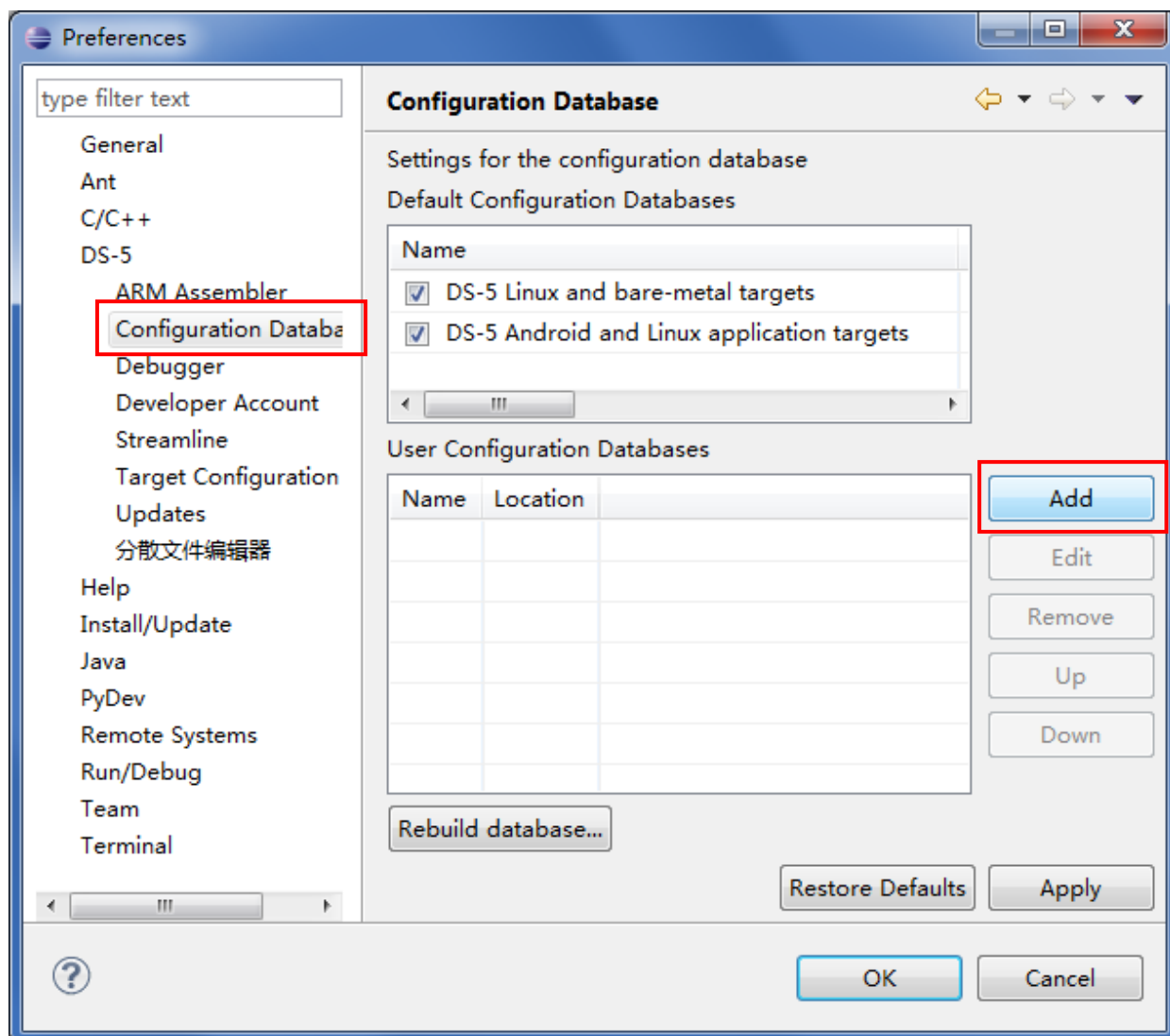
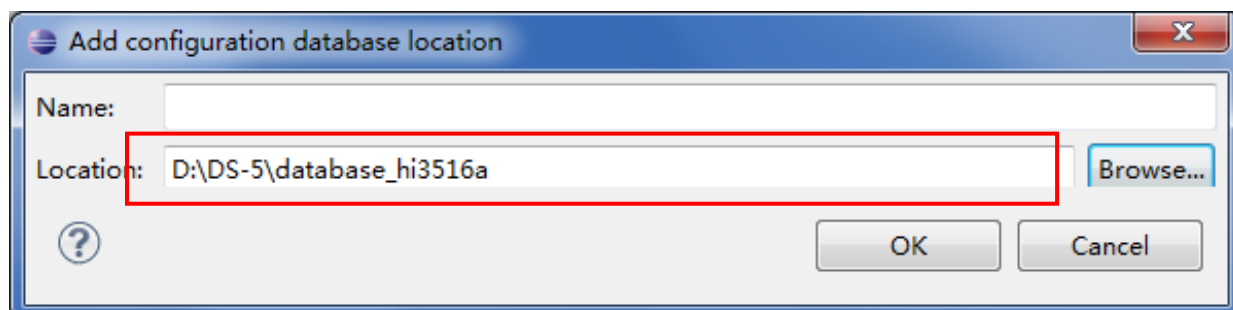


图4-6 Add configure Database 对话框





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. 从主菜单选择【Window】→【Open Perspective】→【DS-5 Debug】，打开 DS-5 Debug 窗口，如图 4-7。
- 步骤 2. 从【Run】菜单选择【Debug Configure】，打开 Debug Configure 窗口，右键点击配置树【DS-5 Debugger】，在弹出的菜单单击【New】创建一个新的配置，如图 4-8。
- 步骤 3. 在名字域内，为新配置键入一个合适的名字，如“Hi3516A_Debug”，如图 4-9。
- 步骤 4. 单击【Connection】标签页配置一个 DS-5 调试器目标连接。此处选择新添加的目标平台配置数据库：【Hisilicon】→【Hi3516A】→【Bare Metal Debug】→【Debug Cortex-A7 via DSTREAM/RVI】，在文本框输入 DS-5 设备的 IP 地址，如图 4-9。
- 步骤 5. 在【Debugger】标签页选中【Connect Only】选项，如图 4-10 所示。
- 步骤 6. 单击【Debug】按钮连接目标平台。

----结束

图4-7 DS-5 Debug 窗口

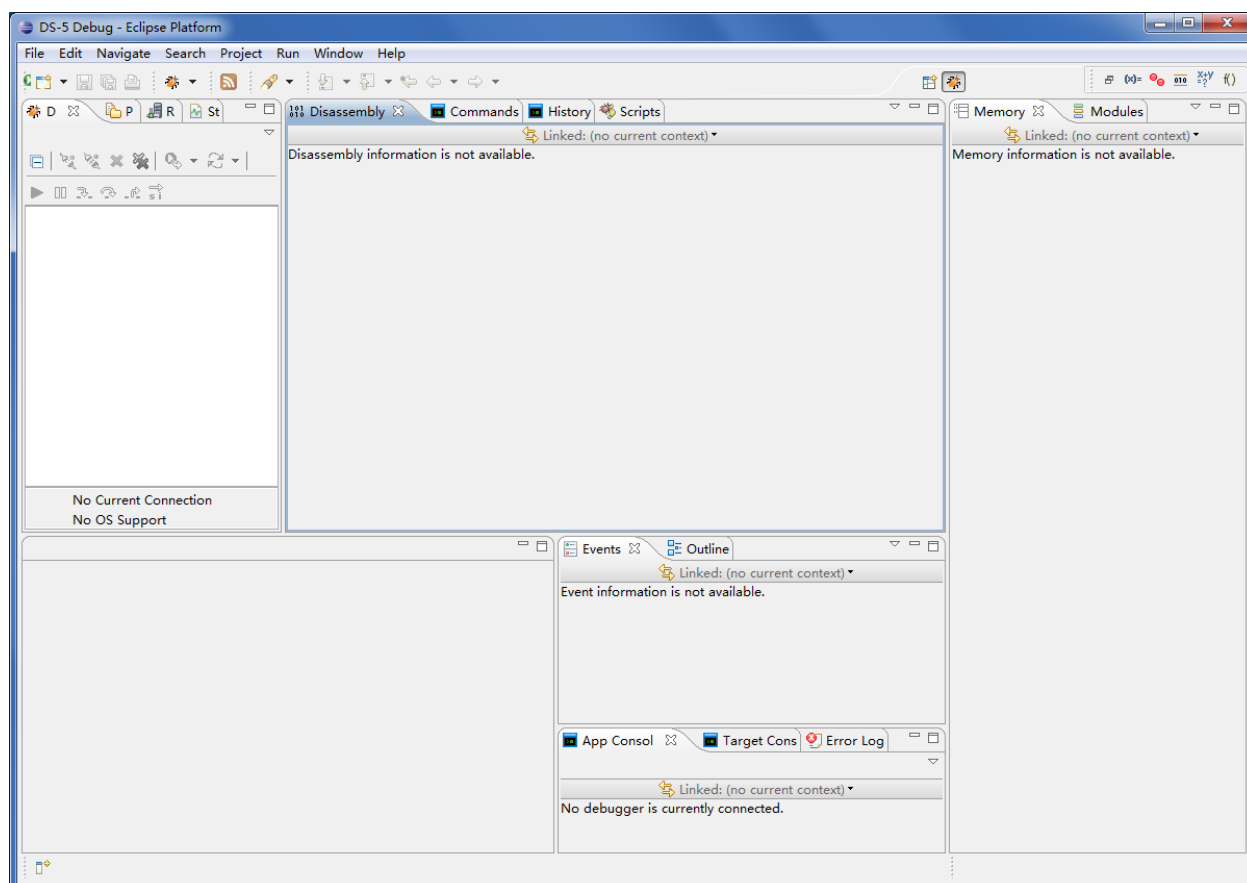




图4-8 Debug Configure 窗口

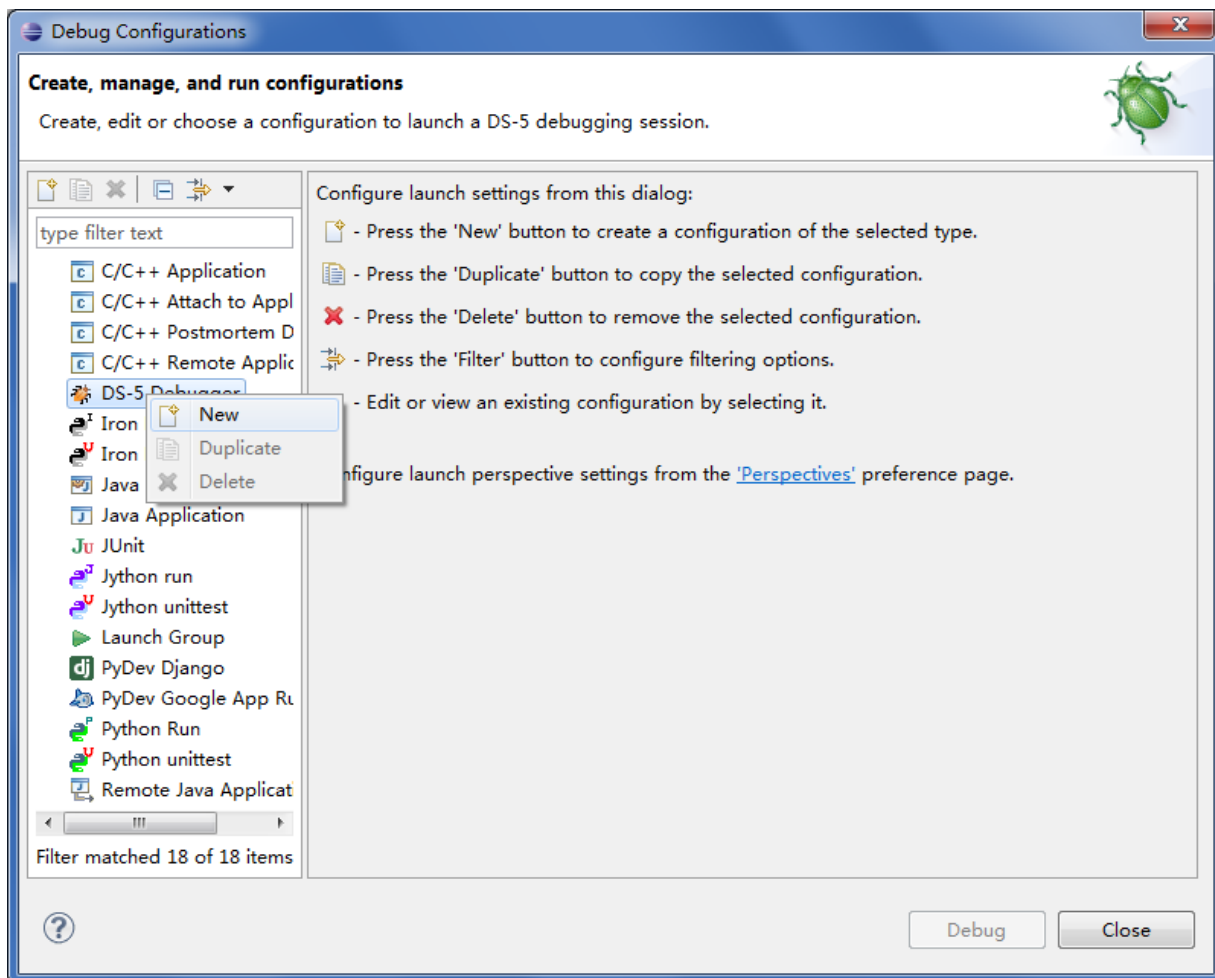




图4-9 Debug Configure 窗口

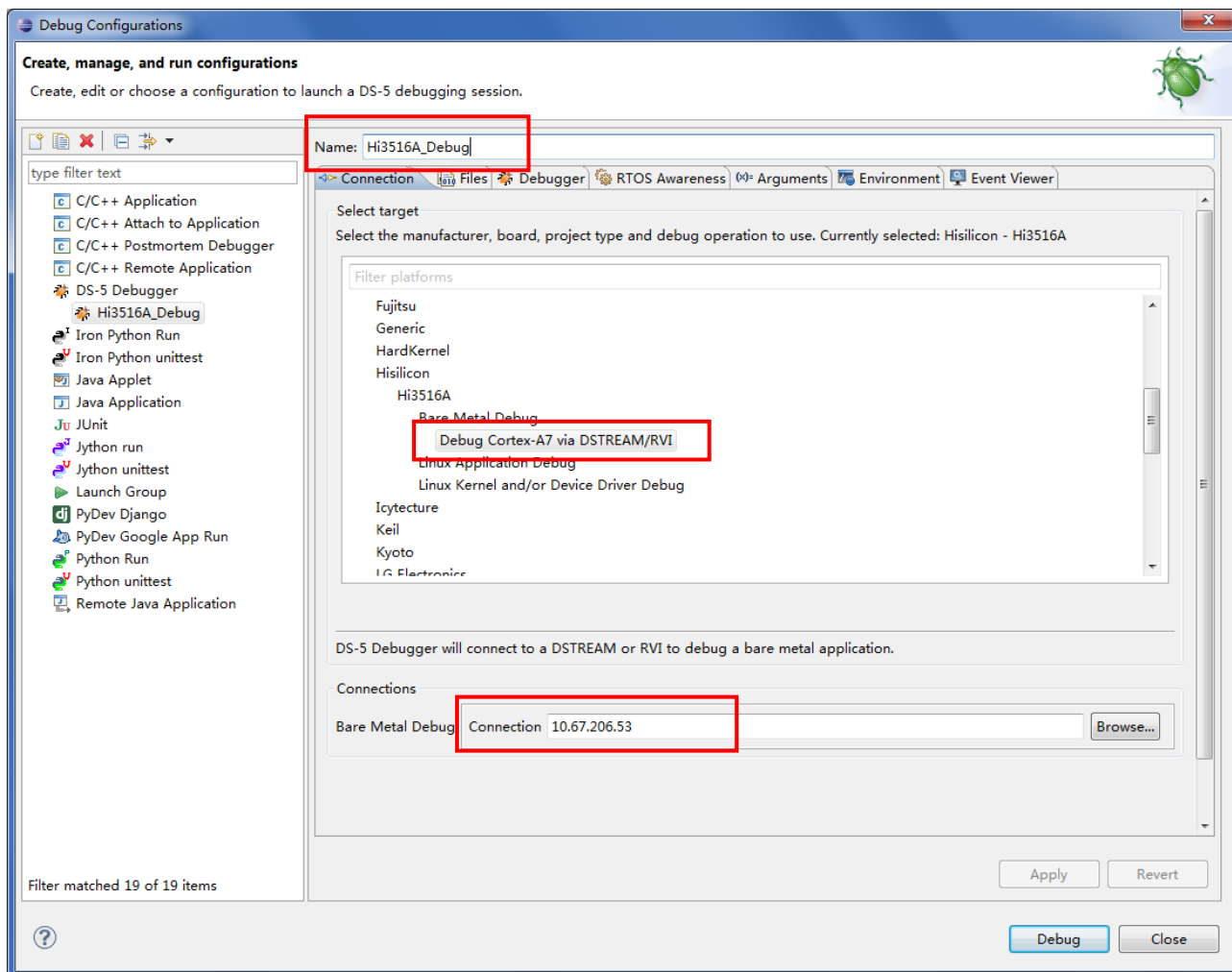
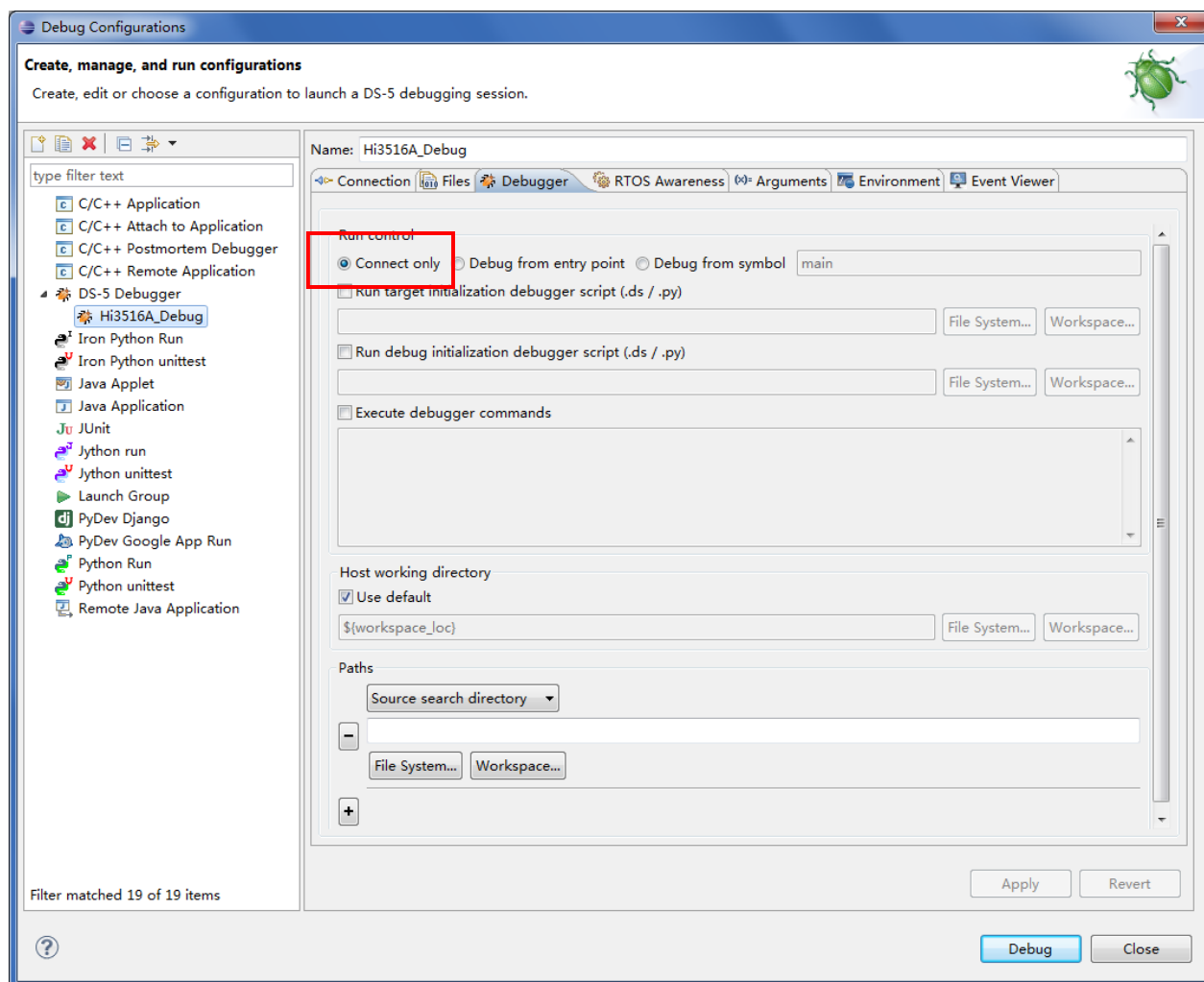







图4-10 Debug Configure 窗口



4.4 使用仿真器烧写启动介质

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 4-11 所示）。

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 0x82000000），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 0x82000000 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 4-12 所示。



图4-11 脚本窗口

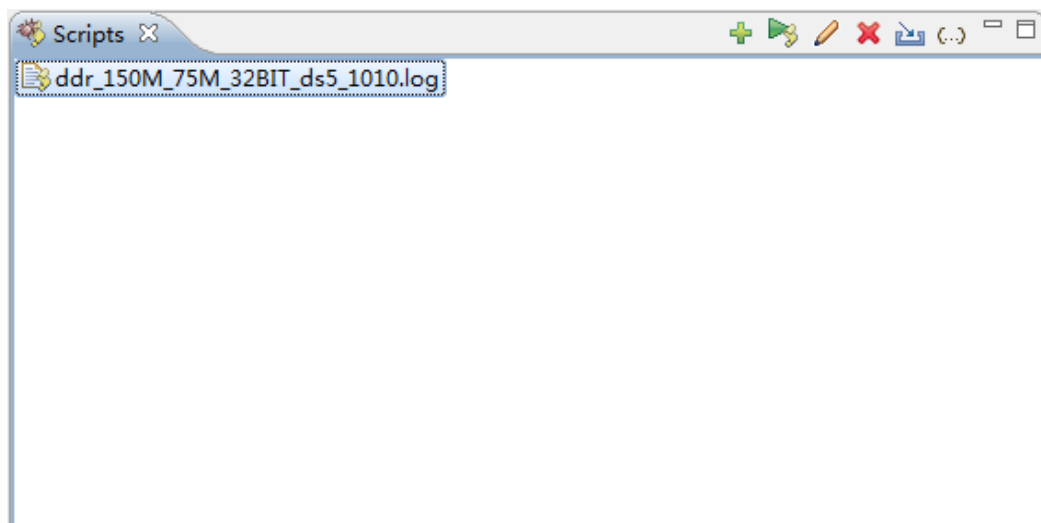
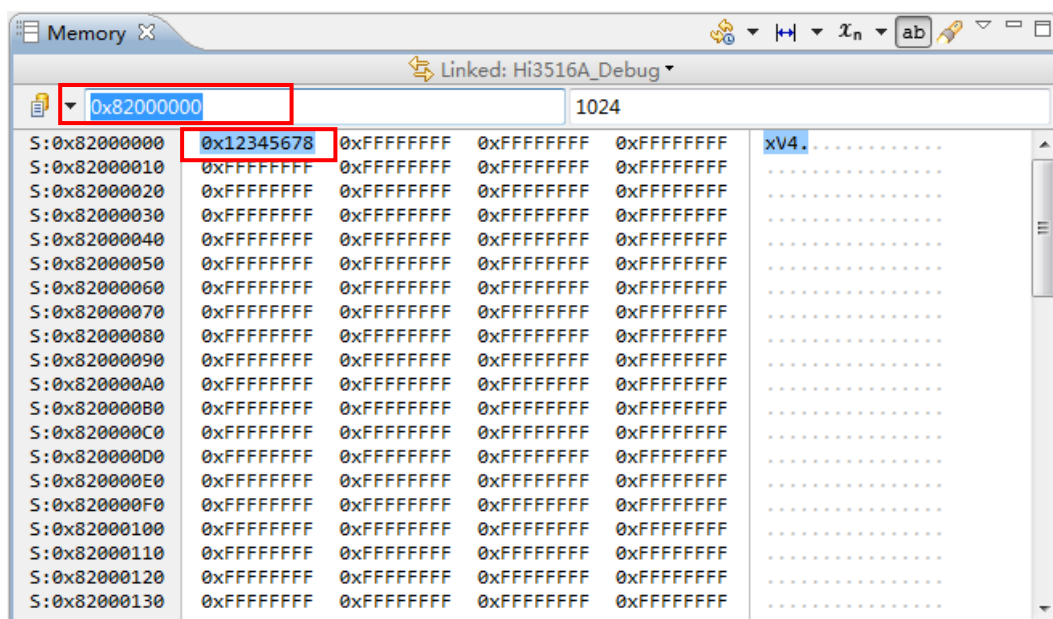


图4-12 Memory 窗口





注意

内存初始化脚本为目录 osdrv\tools\pc\uboot_tools 下的.log 格式文件。

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  弹出图 4-13 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 0x82000000），如图 4-14。
- 步骤 3. 在【Registers】窗口修改 PC 指针值为 0x82000000，如图 4-15 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4-13 Memory 窗口

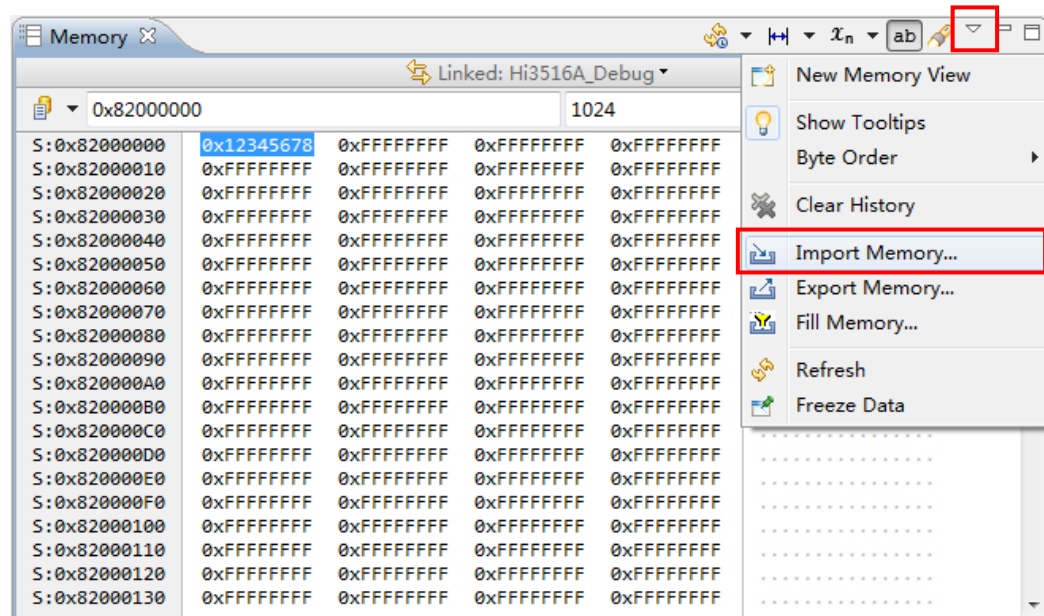




图4-14 Memory Importer 窗口

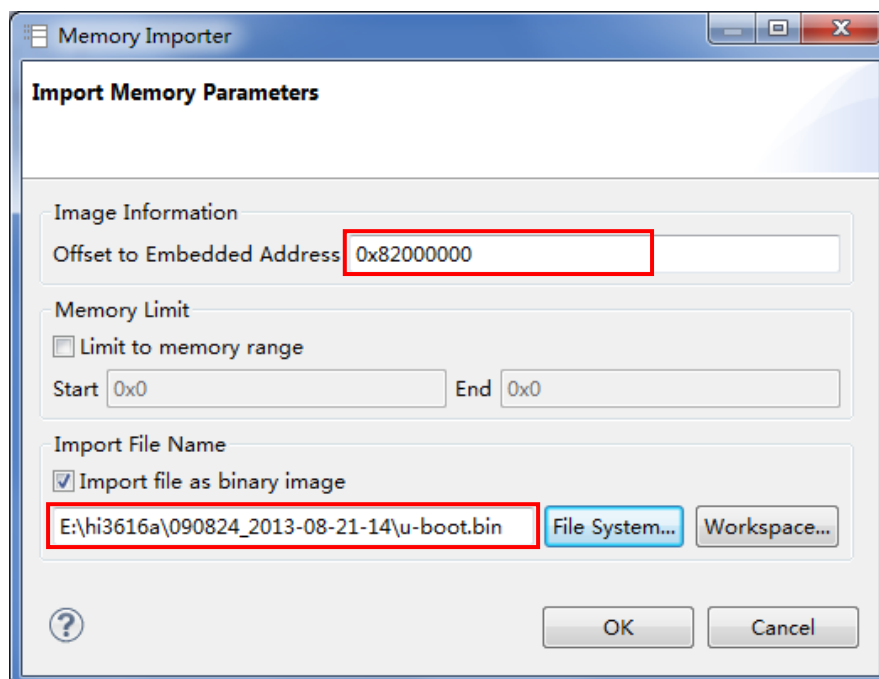
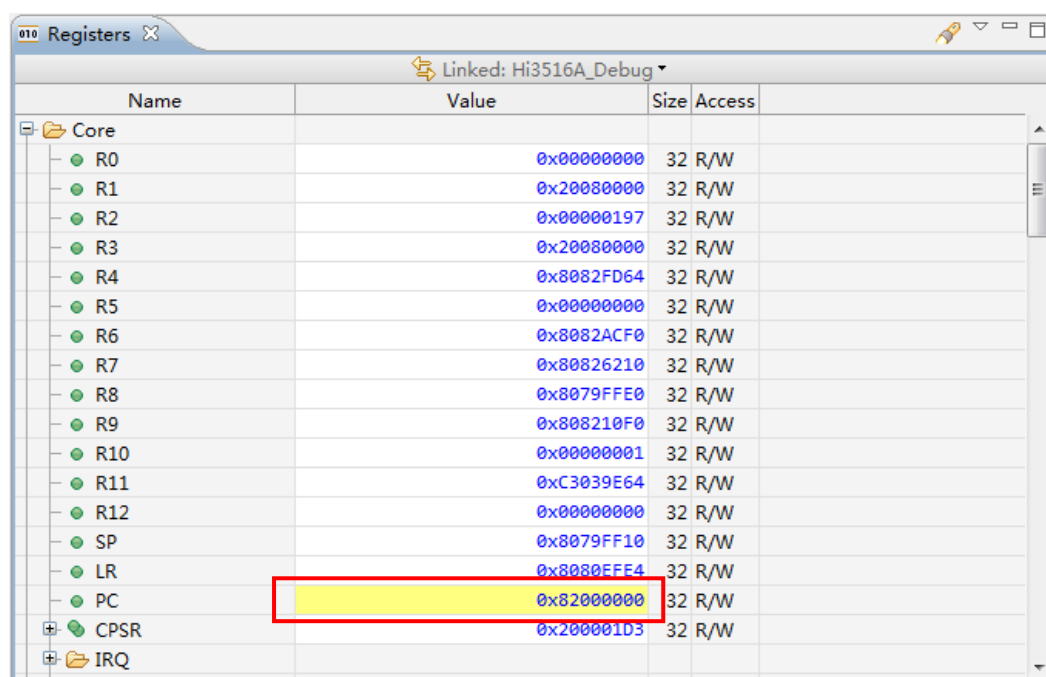


图4-15 Registers 窗口



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：

```
hisilicon# sf probe 0          /*探测并初始化 SPI NOR flash*/
hisilicon# sf erase 0 100000    /*擦除 1M 大小*/
hisilicon# sf write 82000000 0 100000 /*从内存写入 SPI NOR Flash*/
hisilicon# reset               /*重启单板*/
```



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。(其中 Hi3559V100、Hi3519V100/Hi3519V101、Hi3518EV20X、Hi3516CV200、Hi3516CV300 平台支持下述操作, Hi3516A 不支持下述操作)

- 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 #
```