Hi3535 U-boot 移植应用

开发指南

文档版本 00B01

发布日期 2013-08-31

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 Hi3535 单板上如何移植和烧写 U-boot（Hi3535 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
Hi3535 芯片	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
2013-08-31	00B01	第 1 次临时版本发布。

目 录

插图目录

表格目录

1 概述

1.1 概述

Hi3535 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/hi3535	Hi3535 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

在 Hi3535 DMEB 上所选用的外围芯片型号如下：

- DDR SDRAM：K4B2G1646C-HCK0
- NAND Flash：TC58NVG1S3ETA00
- SPI Flash：MX25L12835F

如果选用的外围芯片不是以上型号时，需要适当修改 SDK 中的“**osdrv/tools/pc_tools/uboot_tools/**”目录下的配置表格，对应的单板才能正常运行。

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：

```
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- hi3535_config
```

编译成功后，将在 U-boot 目录下生成 u-boot.bin。

说明

其中 CROSS_COMPILE 表示工具链。文档中统一以 CROSS_COMPILE=arm-hisiXXX-linux-来表示两种情况。

- Hi3535_V100R001C01SPCxxx 对应 uclibc，使用 uclibc 工具链时，CROSS_COMPILE=arm-hisiv100nptl-linux-。
- Hi3535_V100R001C02SPCxxx 对应 glibc，使用 glibc 工具链时，CROSS_COMPILE=arm-hisiv200-linux-。

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_tools/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“ddrc0_init”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标签页“multiplex”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”，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 SDK 中的“osdrv/tools/pc_tools/uboot_tools/”目录下，执行命令：

```
mkboot.sh reg_info.bin u-boot-hi3535.bin
```

其中 **u-boot-hi3535.bin** 就是能够在单板上运行的 U-boot 镜像。

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 RVDS 工具进行烧写。由于芯片特性，在使用 RVDS 时必须要对存储器和芯片进行初始化。在 Hi3535 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《Fastboot 工具使用说明 Application Notes》。

3.3 两种 Flash 的 U-boot 烧写方法

3.3.1 SPI Flash 烧写方法

SPI Flash 烧写方法如下：

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-hi3535.bin    /*U-boot下载到内存*/
hisilicon# sf probe 0                   /*探测并初始化SPI flash*/
hisilicon# sf erase 0x0 0x100000            /*擦除 1M大小*/
hisilicon# sf write 0x82000000 0x0 0x100000   /*从内存写入SPI Flash*/
```

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 烧写方法

NAND Flash 烧写方法如下：

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-hi3535.bin       /*U-boot下载到内存*/
hisilicon# nand write 0x82000000 0 100000   /*从内存写入NAND Flash*/
```

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

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：

- RealView -ICE 仿真器
- RealView Debugger4.0

4.2 ARM 调试工具简介

4.2.1 RealView -ICE 仿真器

RealView -ICE 仿真器是一款高性能实时仿真器，该款仿真器支持 ARM 内核结构（如 ARM7、ARM9 Xscale、ARM11、ARM_CORTEXA9）的处理器，是进行基于 ARM 开发调试的必备工具。

RealView -ICE 仿真器的特点如下：

- 支持内含 Embedded ICETM logic 的 ARM 内核芯片，同时也支持尚在开发中的 ARM 内核芯片，包括 ARM7、ARM710、ARM720、ARM740、ARM9、ARM920、ARM10、ARM11 等 ARM 内核调试。
- 支持目标系统的电源从 1.8V～5V 自适应。
- 通过网口连接仿真器，用户可以轻松修改寄存器、存储器，设置硬件断点，增加观察窗口等。可以在二进制文件的开始位置或者结束位置保存自定义的数据和调色板。

4.2.2 RealView Debugger4.0

RealView Debugger4.0 是由 ARM 公司提供的软件开发调试软件，为软件开发者开发高质量的 ARM 代码。RealView Debugger 提供了如下便利：

- 支持简单和复杂的断点设置。
- 提供观察窗口，可查看变量的变化情况。
- 支持所有新的和已经存在的 ARM 处理器。

- 增强的 Windows 管理模式。
- 增强的数据显示、修改和编辑。
- 提供完备的命令行接口。

4.3 使用 ARM 调试工具

要使用 Multi-ICE 进行程序调试或者向开发板烧写 U-boot 程序，首先必须启用 Multi-ICE server 程序查找到当前硬件连接的 ARM 芯片，然后才能通过 RealView Debugger 调试软件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 RealView Debugger。

1. 安装 ARM developer Suite v4.0。ARM developer Suite v4.0 是由 ARM 公司提供的 RealView Debugger 安装程序。安装前，请先阅读 ARM 的相关文档。
2. 运行 RealView Debugger。运行后，RealView Debugger 启动如图 4-1 所示。
3. 连接 RealView-ICE。单击“Selects targets to connect to”，弹出如图 4-2 所示的窗口。单击 RealView-ICE 后的 add，查找并连接仿真器，如图 4-3 所示。
4. 连接 A9。单击图 4-4 中的“Auto Configure”按钮，可扫描到 A9，出现如图 4-5 所示的窗口。然后保存并关闭，在 RealView ICE 下面就能看到目标 Cortex-A9_0，再双击连接就可以进行调试，如图 4-6 所示。

----结束

图4-1 RealView Debugger 窗口

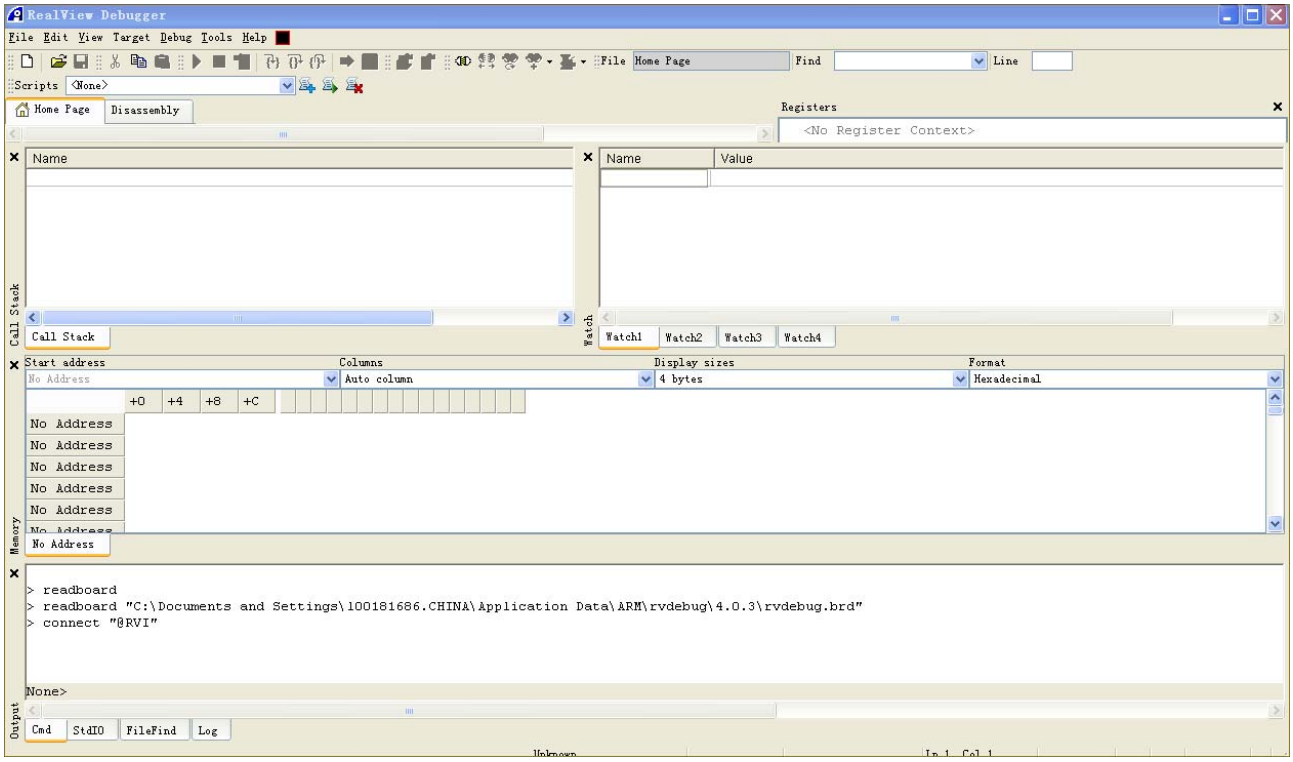


图4-2 Connetct to Target 窗口

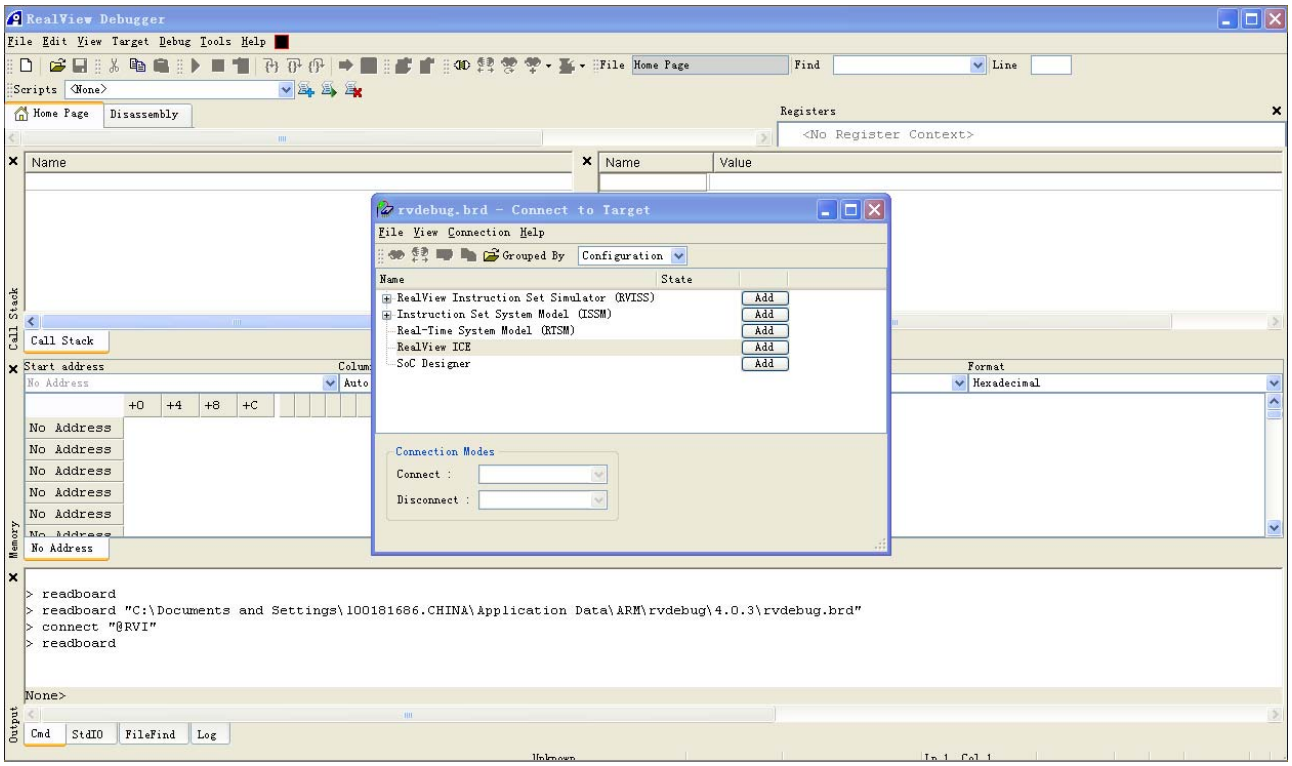


图4-3 Choose RealView-ICE

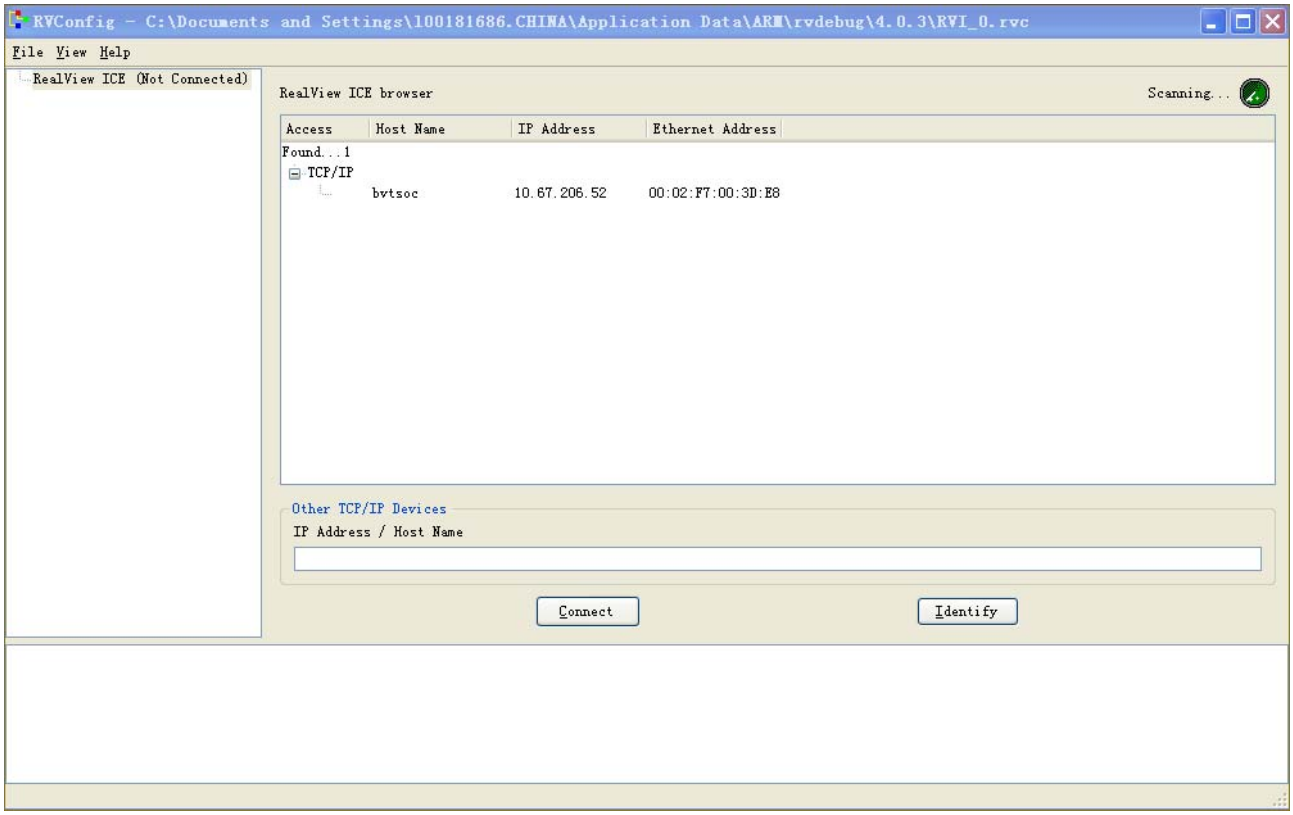


图4-4 RealView Debugger 信息提示框

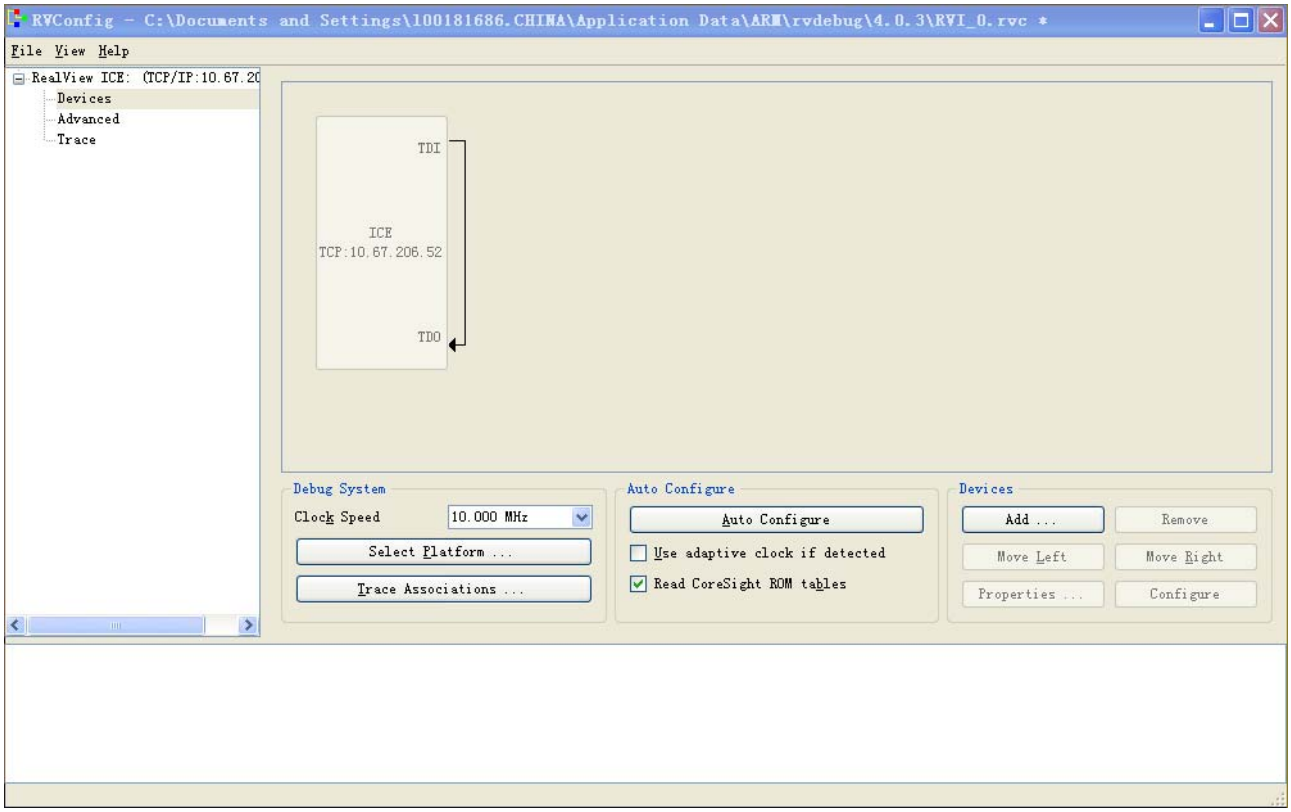


图4-5 RealView Debugger 信息提示框 2

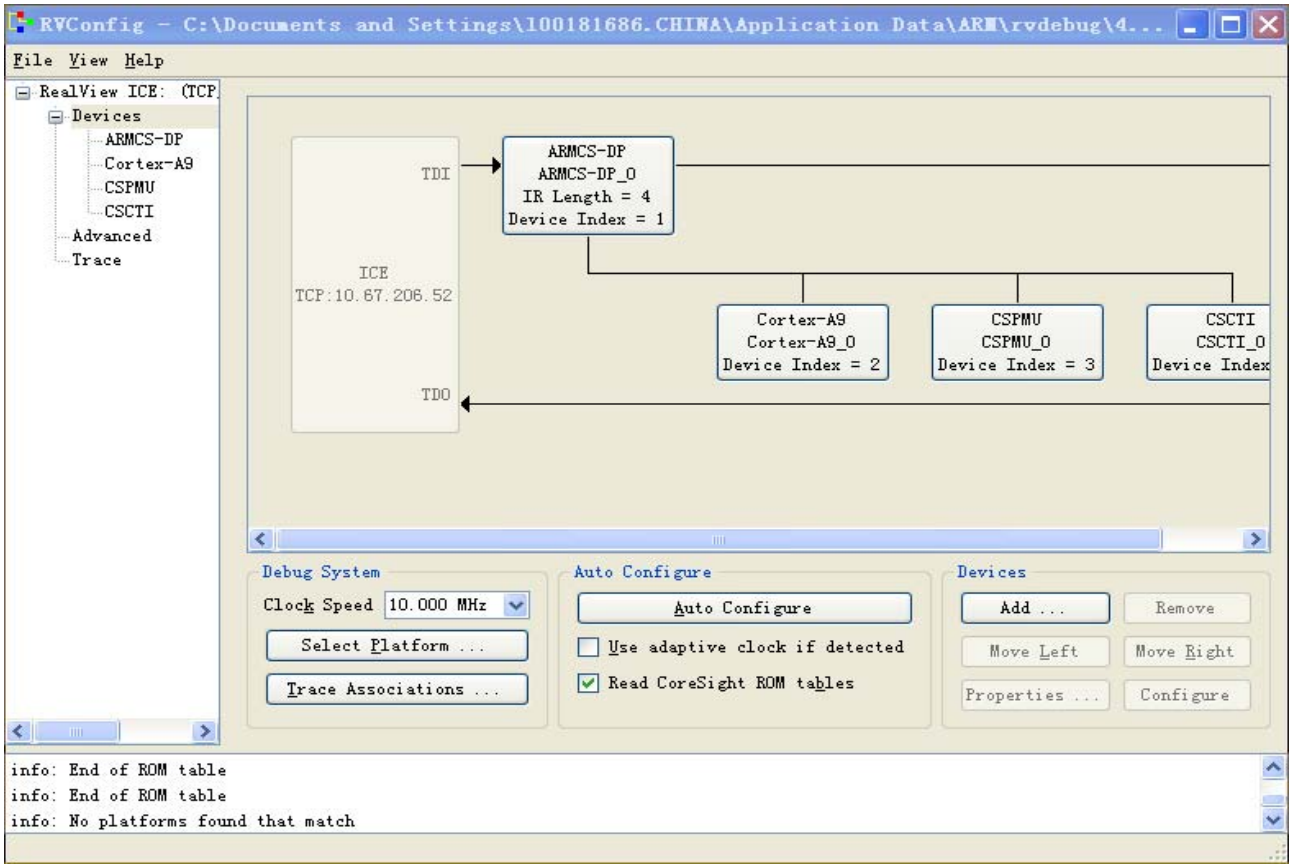
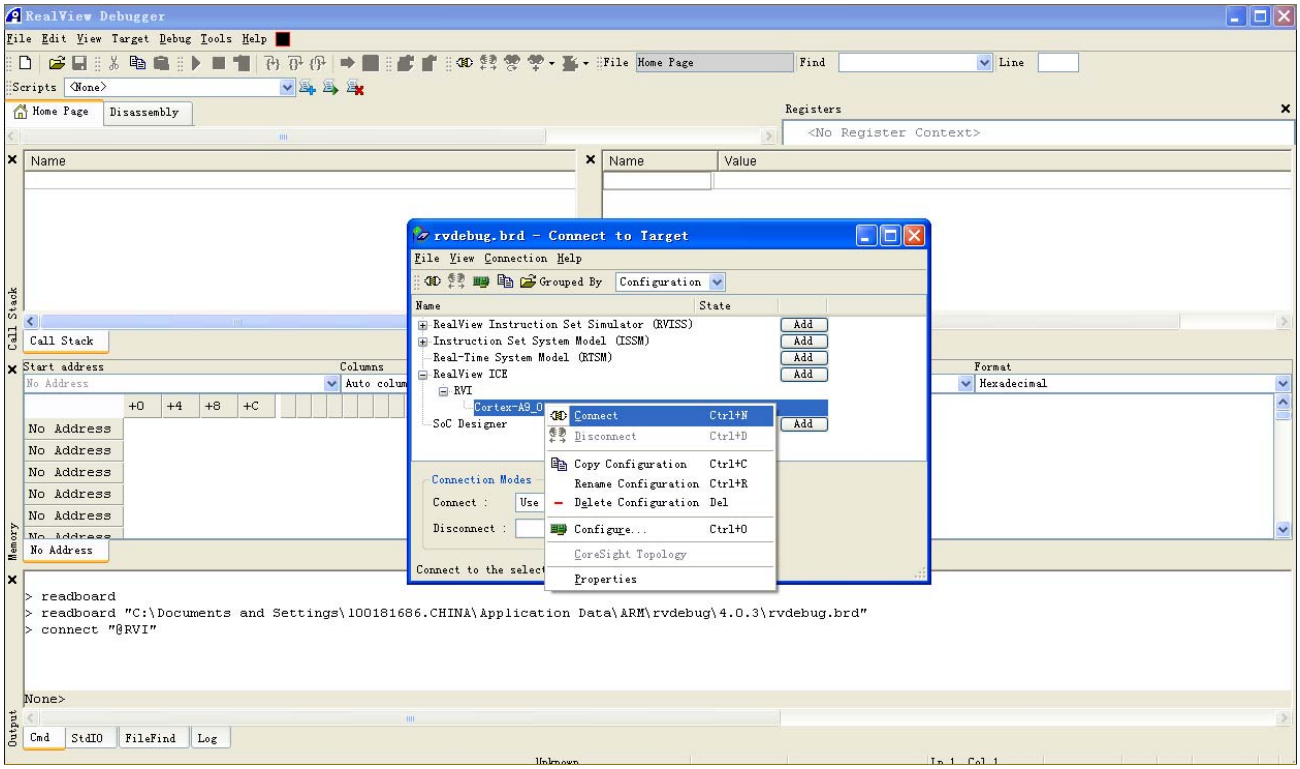


图4-6 Connect to A9



4.4 使用仿真器烧写 Flash

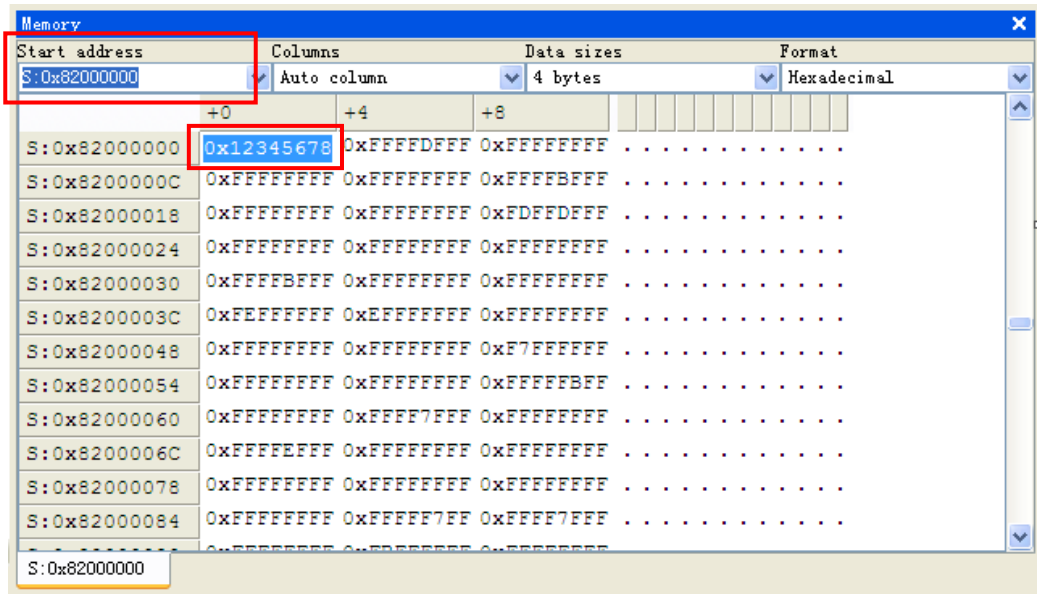
4.4.1 内存初始化

单击菜单 Tools->Include commands from File…运行内存初始化脚本（如果此时仿真器处于运行状态，则需单击按钮“stop execution on target”暂停仿真器）。

可通过以下方式验证内存初始化成功与否：

在 Memory 窗口“Start address”栏中输入想查看的内存地址信息（如 0x82000000），回车后查看表格是否显示 32 位内存的值。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 4-7 所示。）

图4-7 Memory 窗口



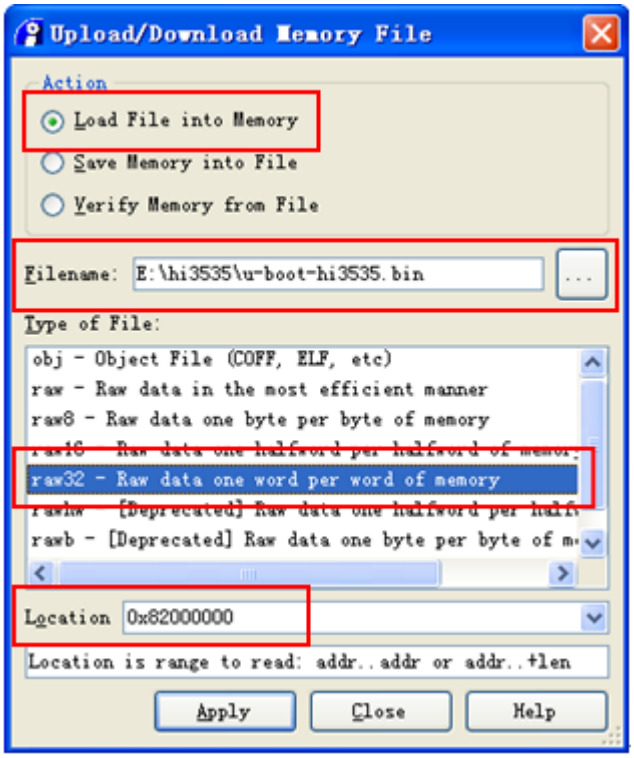
注意

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

4.4.2 下载 U-Boot 映像

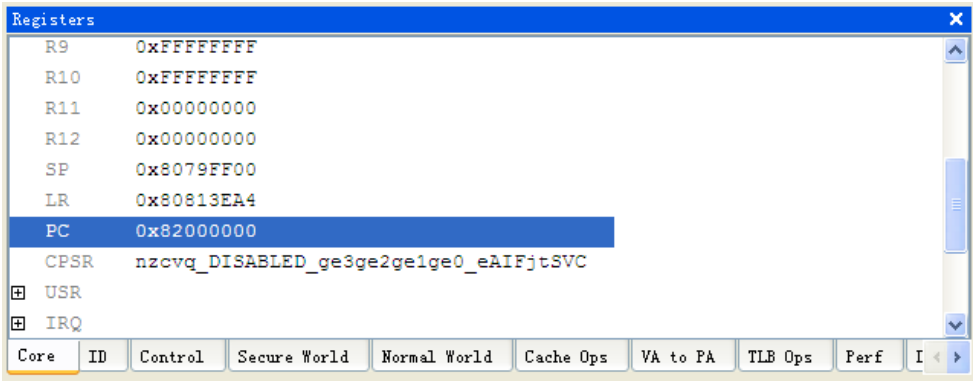
单击菜单 Debug>>Memory/Regiter Operations，下载 u-boot 映像到内存地址（如 0x82000000），如图 4-8 所示。

图4-8 映像下载窗口



在 Regiters 窗口修改 PC 为 0x82000000，如图 4-9 所示。单击按钮“Run current program counter”启动 U-Boot，此时可通过串口查看 U-Boot 启动信息。

图4-9 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 Flash 中。

以 SPI Flash 为例，其烧写步骤如下：

hisilicon# sf probe 0 /*探测并初始化 SPI flash*/

hisilicon# sf erase 0 100000 /*擦除 1M 大小*/

hisilicon# sf write 82000000 0 100000 /*从内存写入 SPI Flash*/

hisilicon# reset /*重启单板*/

----结束