Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C U-boot Porting

# Development Guide

**Issue** 04

**Date** 2013-07-31

HiSilicon Technologies Co., Ltd.

| | |
|---|---|
| Address: | Huawei Industrial Base<br>Bantian, Longgang<br>Shenzhen 518129<br>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 Hi3521 board) on the Hi3521/Hi3520A/Hi3520D/Hi3515A/Hi3515C board, and how to using ARM debugging tools.

## Related Version

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

| Product Name | Version |
|---|---|
| Hi3521 | V100 |
| Hi3520A | V100 |
| Hi3520D | V100 |
| Hi3515A | V100 |
| Hi3515C | 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 04 (2013-07-31)

This issue is the fourth official release, which incorporates the following changes:

**Chapter 2 Porting the U-boot**

In section 2.5, the caution is updated.

## Issue 03 (2013-06-21)

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

The Hi3515C compilation code is added.

## Issue 02 (2013-04-03)

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

**Chapter 2 Porting the U-boot**

A caution related to the Hi3520D and Hi3515A is added.

## Issue 01 (2013-02-28)

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

**Chapter 1 Overview**

The Hi3520D compilation code is added.

**Chapter 2 Porting the U-boot**

The Hi3520D compilation code is added.

**Chapter 3 Burning the U-boot**

In section 3.3.2, the caution "Only the Hi3521 and Hi3520A support the NAND flash" is added.

## Issue 00B20 (2012-12-30)

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

**Chapter 2 Porting the U-boot**

In section 2.2, the command for compiling the U-boot is modified.

## Issue 00B10 (2012-06-30)

This issue is the first official release, which incorporates the following changes:

**Chapter 2 Porting the U-boot**

In section 2.5, the final image name is changed, ensuring that the image name is the same as that in the SDK.

**Chapter 4 Using ARM Debugging Tools**

Figure 4-1 to Figure 4-6 are changed.

## Issue 00B02 (2012-06-08)

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

**Chapter 2 Porting the U-boot**

In section 2.1, the models of the peripherals on the Hi3521 board are changed.

In section 2.3, the compilation toolchain is changed from arm-hisivX00-linux- to arm-hisiv100nptl -linux- and arm-hisiv200-linux-.

## Issue 00B01 (2012-04-20)

This issue is the first draft release.

# Contents

# Figures

# Tables

# 1 Overview

**NOTE**
The Hi3521 is used as an example when the conditions apply to the Hi3521, Hi3520A, Hi3520D, Hi3515A, and Hi3515C. Their differences are separately described.

## 1.1 Description

The bootloader of the Hi3521 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/godarm/ | Indicates the code of HiSilicon boards. |
| lib_xxx | 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. |

| Directory | Description |
|---|---|
| arch | Indicates the code of the chip architecture and the entry code of the U-boot. |
| 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 Overview

The following are the peripherals on the Hi3521 board:

- Double-data rate (DDR) synchronous dynamic random access memory (SDRAM): K4B2G1646C-HCK0
- NAND flash: TC58NVG1S3ETA00
- Serial peripheral interface (SPI) flash: FL128PIFL121QQ030

If other peripherals are used, the corresponding board can work properly only after you modify the configuration sheet in **osdrv/ tools/pc_tools/uboot_tools/** of the SDK.

## 2.2 Compiling the U-boot

After preceding operations are complete, you can compile the U-boot by running the following commands:

```
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- godarm_config //Select the
Hi3521
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- godcare_config //Select
Hi3520A
make ARCH=arm CROSS_COMPILE=arm-hisivXXX-linux- hi3520d_config  //Select the
Hi3520D/Hi3515A
```

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

**NOTE**

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

- Hi35xx_V100R001C01SPCxxx corresponds to uclibc. If the uclibc tool chain is used, the **CROSS_COMPILE** parameter is set to **arm-hisiv100nptl-linux-**.
- Hi35xx_V100R001C02SPCxxx corresponds to glibc. If the glibc tool chain is used, the **CROSS_COMPILE** parameter is set to **arm-hisiv200-linux-**.

**CAUTION**

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

## 2.3 Configuring the DDR

In Windows, open the configuration sheet in **osdrv/ tools/pc_tools/uboot_tools/** of the SDK. If different DDR SDRAM is used, modify the contents on the tab pages of the configuration sheet based on the component features.

## 2.4 Configuring Pin Multiplexing

If the pin multiplexing relationship changes, modify the contents on the related tab page of the configuration sheet.

## 2.5 Generating the Final U-boot Image

Perform the following steps:

**Step 1** Save settings after modifying the configuration sheet.

**Step 2** Click **Generate reg bin file** on the first tab page of the configuration sheet to generate the temporary file **reg_info.bin**.

**Step 3** Copy **u-boot.bin** (it is generated after the U-boot is compiled) and **reg_info.bin** to **osdrv/uboot/ uboot_tools** of the SDK, and run the following command:

```
mkboot.sh reg_info.bin u-boot-hi3521_930MHz.bin
```

**u-boot-hi3521_930MHz.bin** is the final u-boot image that can run on the board.

**----End**

**CAUTION**

To create U-boot images of the Hi3520D, Hi3515A, and Hi3515C, generate two temporary files by using Hi3520D, Hi3515A, and Hi3515C configuration sheets, rename the temporary files **reg_info_hi3520d.bin** and **reg_info_hi3515a.bin** respectively, copy **reg_info_hi3520d.bin**, **reg_info_hi3515a.bin**, and compiled **u-boot.bin** to **osdrv/tools/pc_tools/uboot_tools/** of the SDK, and then run **mkboot-hi3520d.sh reg_info_hi3520d.bin reg_info_hi3515a.bin u-boot-hi3520d.bin**. Note that **reg_info_hi3520d.bin** must be before **reg_info_hi3515a.bin** in this command.

A 4800-byte blank zone must be reserved in the U-boot code segment for storing **reg_info_hi3520d.bin** and **reg_info_hi3515a.bin**. Each of them occupies 2400 bytes. The blank zone should not be greater than 4800 bytes due to the limitation of the on-chip RAM. Otherwise, the fastboot cannot be burnt. If a blank zone greater than 4800 bytes is required, you are advised to burn a U-boot image that is less than or equal to 4800 bytes by using the fastboot tool, and then burn a U-boot image greater than 4800 bytes over the network to replace the previously burnt one.

To combine **reg_info.bin** of the Hi3520D, Hi3515A, or Hi3515C with the U-boot image, you need to comment out the macro **ENABLE_HI3520D_BLANK** or **ENABLE_HI3515A_BLANK** in **./include/configs/hi3520d.h** of the U-boot.

The **REG_INFO_BLANK_SIZE** macro specifies the size of the blank zone reserved for the temporary file (**reg_info_xxx.bin**). The default value of **REG_INFO_BLANK_SIZE** is 2400, and you can change it.

To combine more temporary files with the U-boot image, perform the following steps:

**Step 1** Modify the code in rows 54−61 in **./arch/arm/cpu/hi3520d/start.S** as follows:

```
__blank_zone_start:
#ifdef ENABLE_HI3520D_BLANK
.fill REG_INFO_BLANK_SIZE,1,0     /*Blank zone*/
#endif
#ifdef ENABLE_HI3515A_BLANK
.fill REG_INFO_BLANK_SIZE,1,0     /*Blank zone*/
#endif
/*Add new blank zones.*/
__blank_zone_end:
```

**Step 2** Modify **mkboot-hi3520d.sh** in **osdrv/tools/pc_tools/uboot_tools/** as follows:

```
dd if=$1 of=./fb2 bs=2400 conv=sync
dd if=$2 of=./fb3 bs=2400 conv=sync
/*You can add commands similar to the preceding two commands.*/
dd if=./u-boot.bin of=./fb4 bs=1 skip=4864 /*4864 = 64 + Size of all blank zones*/
cat fb1 fb2 fb3 fb4 > $3
```

**----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 the U-boot by using the RealView Development Suite (RVDS) tool over the Ethernet port. According to the chip feature, you must download the scripts for initializing the DDR and chip to the board by using the RVDS tool. The initialization scripts are provided in the Hi3521 SDK. When different DDRs are used, the Hi3521 can work properly only when the initialization scripts are reconfigured.

## 3.2 Burning the U-boot by Using the BOOTROM

For details, see the *Fastboot 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 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     /*Set the DDR value of all Fs.*/
hisilicon# tftp 0x82000000 u-boot-hi3521_930MHz.bin /*Download the U-boot to
the DDR.*/
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 DDR
to the SPI flash.*/
```

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

**----End**

### 3.3.2 Burning the U-boot to the NAND Flash

**CAUTION**

Only the Hi3521 and Hi3520A support the 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 0 100000           /*Erase 1 MB capacity of the NAND
flash.*/
hisilicon# mw.b 0x82000000 ff 100000       /*Set the DDR value to all Fs.*/
+hisilicon# tftp 0x82000000 u-boot-hi3521_930MHz.bin    /*Download the
U-boot to the DDR.*/
hisilicon# nand write 0x82000000 0 100000   /*Write the U-boot from the DDR
to the NAND flash.*/
```

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

**----End**

# 4 Using ARM Debugging Tools

## 4.1 Overview

This chapter describes how to use the following tools to debug the ARM processor:

- RealView -ICE Simulator
- RealView Debugger4.0

## 4.2 ARM Debugging Tools

### 4.2.1 RealView -ICE Simulator

The RealView-ICE simulator is a high-performance real-time simulator. It supports the processor with the ARM core (such as ARM7, ARM9 Xscale, ARM11, or ARM_CORTEXA9) and is essential for the ARM-based development and debugging.

The RealView-ICE simulator has the following features:

- Supports the debugging of the ARM core chips with the embedded ICETM logic and the debugging of the developing chips with the ARM cores including ARM7, ARM710, ARM720, ARM740, ARM9, ARM920, ARM10, and ARM11.
- Supports the power supplies from 1.8 V to 5 V in the target system.
- Allows you to modify the values of registers and memories, set hardware breakpoints, and add inspection windows over the Ethernet port.
- Allows you to save user-defined data and palettes at the beginning or at the end of binary files.

### 4.2.2 RealView Debugger4.0

The RealView Debugger4.0 is a software development debugging tool provided by ARM. This tool is designed to develop high-quality ARM code for software developers. The RealView Debugger4.0 has the following functions:

- Allows you to set simple or complicated breakpoints.
- Provides inspection windows for checking the change of variables.
- Supports all the new and existing ARM processors.

- Supports the enhanced Windows management mode.
- Provides the enhanced functions of displaying, modifying, and editing data.
- Provides comprehensive command-line interfaces (CLIs).

## 4.3 Usage Guidelines

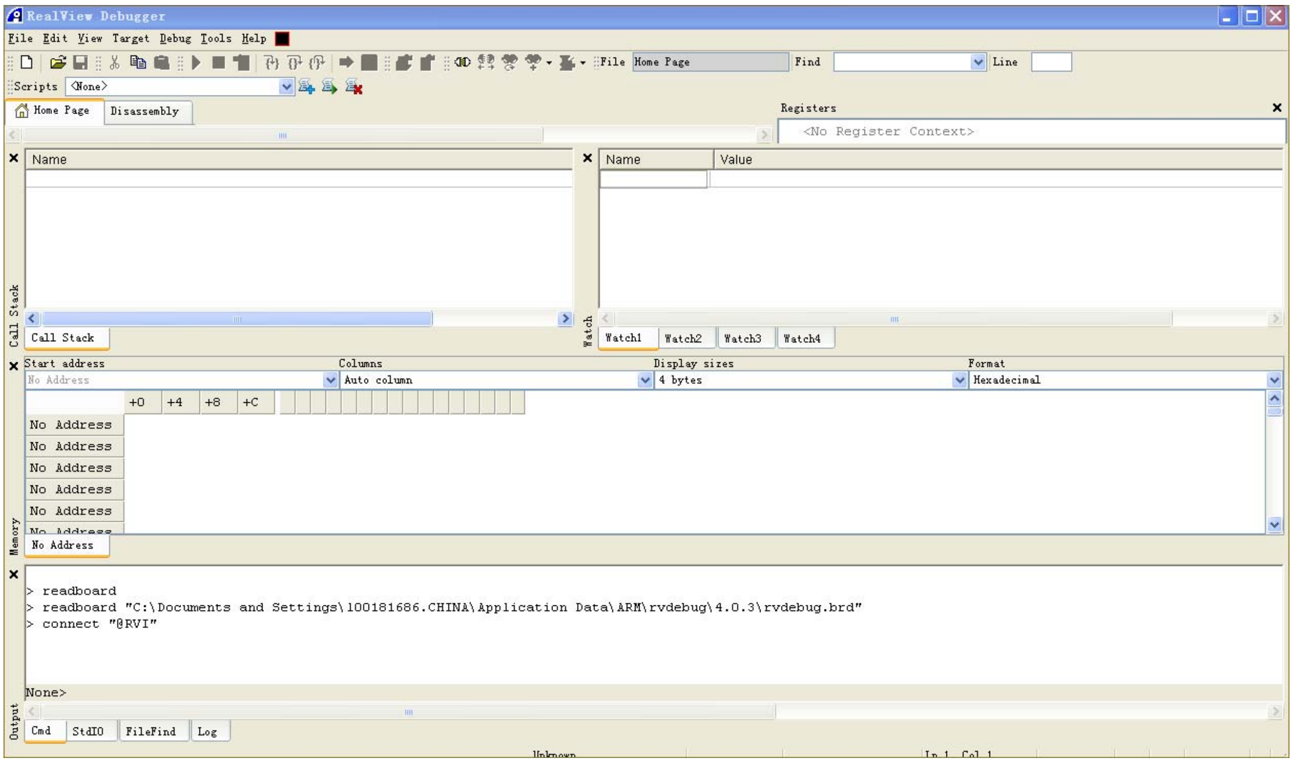
Before debugging programs or burning the U-boot to the board by using the Multi-ICE server, you must run the Multi-ICE server to locate the ARM chip connected to the hardware, and start the RealView Debugger.

For details about how to use ARM debugging tools, see the documents provided by ARM. The following is an example using the RealView Debugger. Perform the following steps:

**Step 1** Install the ARM developer Suite v4.0. It is the RealView Debugger setup program provided by ARM. Before installation, read relevant documents provided by ARM.

**Step 2** Run the RealView Debugger, as shown in Figure 4-1.

**Figure 4-1** RealView Debugger

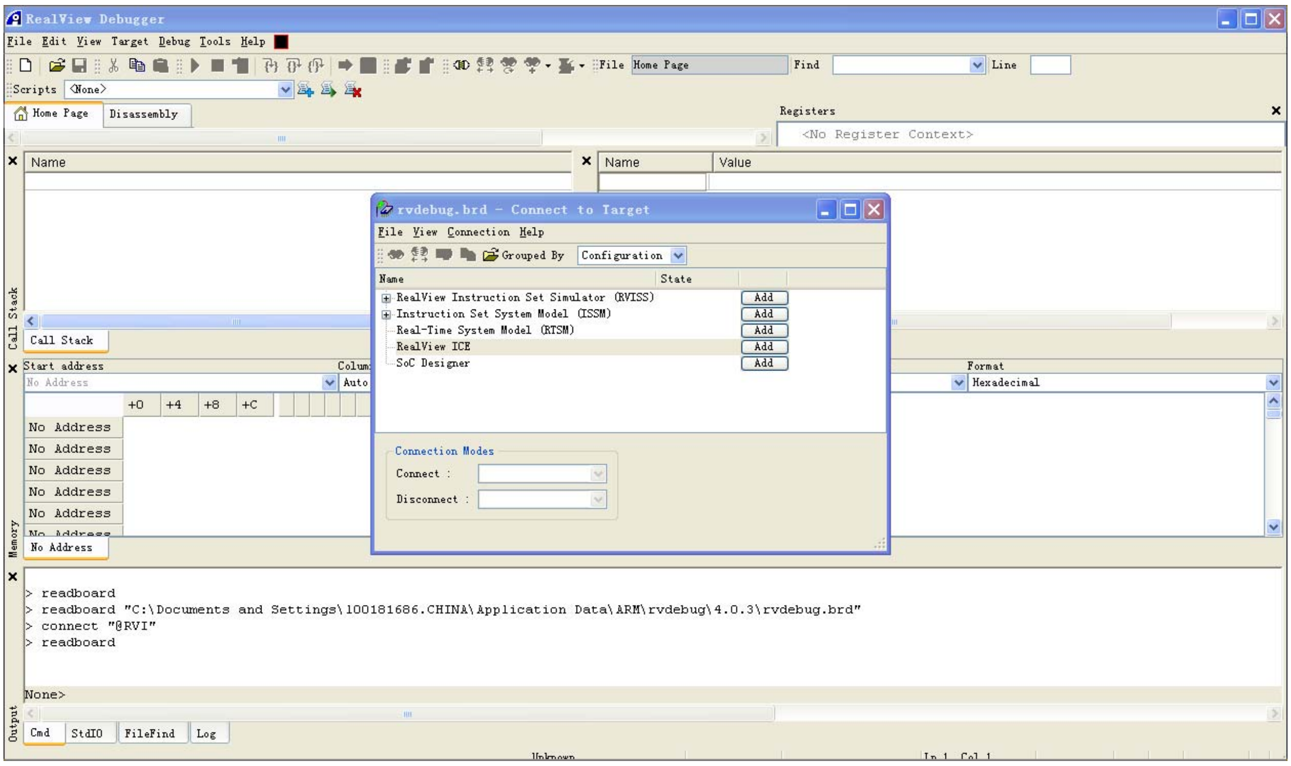


```
> readboard
> readboard "C:\Documents and Settings\100181686.CHINA\Application Data\ARM\rvdebug\4.0.3\rvdebug.brd"
> connect "@RVI"

None>
```

**Step 3** Click to connect the simulator to the RealView-ICE. The window shown in Figure 4-2 is displayed.

**Figure 4-2** Connect to Target window

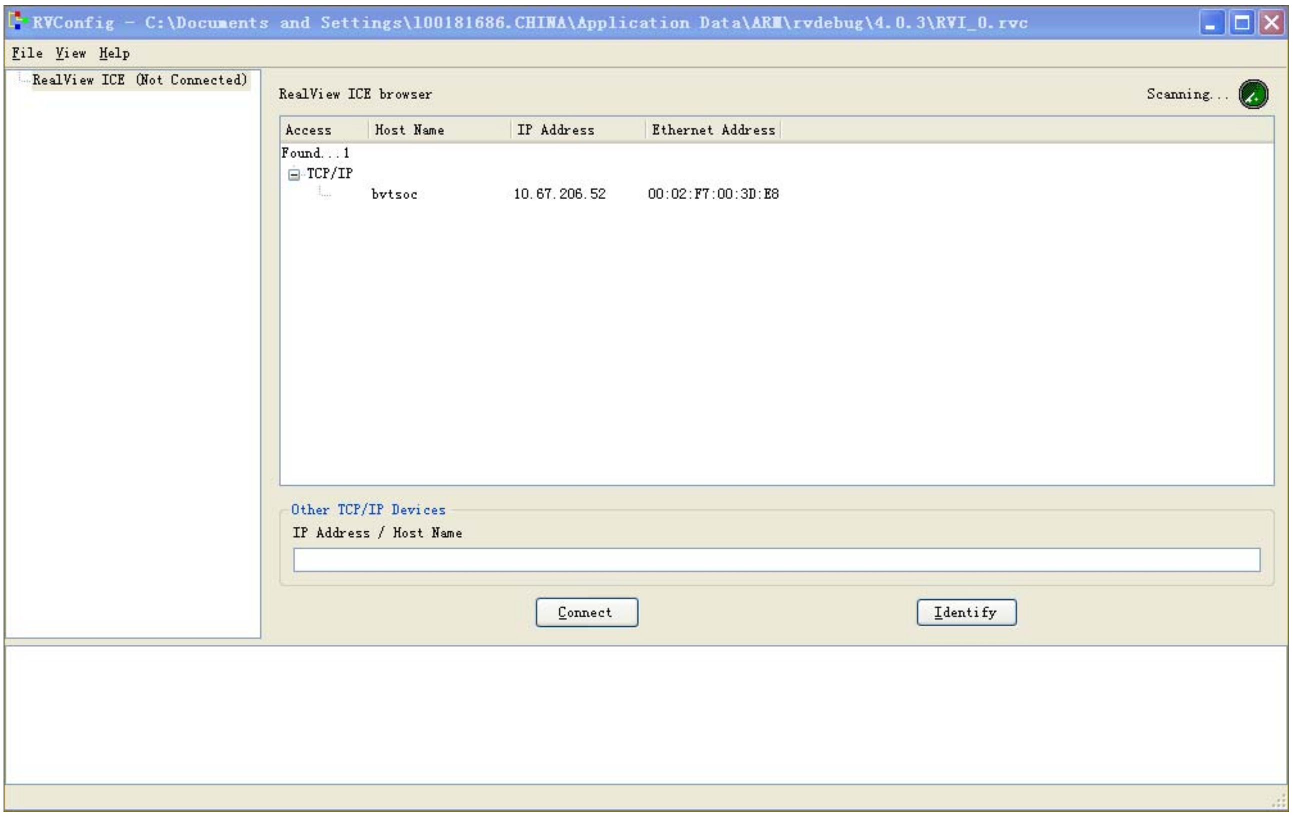


| Name | State | |
|---|---|---|
| RealView Instruction Set Simulator (RVISS) | | Add |
| Instruction Set System Model (ISSM) | | Add |
| Real-Time System Model (RTSM) | | Add |
| RealView ICE | | Add |
| SoC Designer | | Add |

```
> readboard
> readboard "C:\Documents and Settings\100181686.CHINA\Application Data\ARM\rvdebug\4.0.3\rvdebug.brd"
> connect "@RVI"
> readboard
```

**Step 4** Click **Add** after **RealView-ICE**, locate a simulator, and connect it to the RealView-ICE, as shown in Figure 4-3.

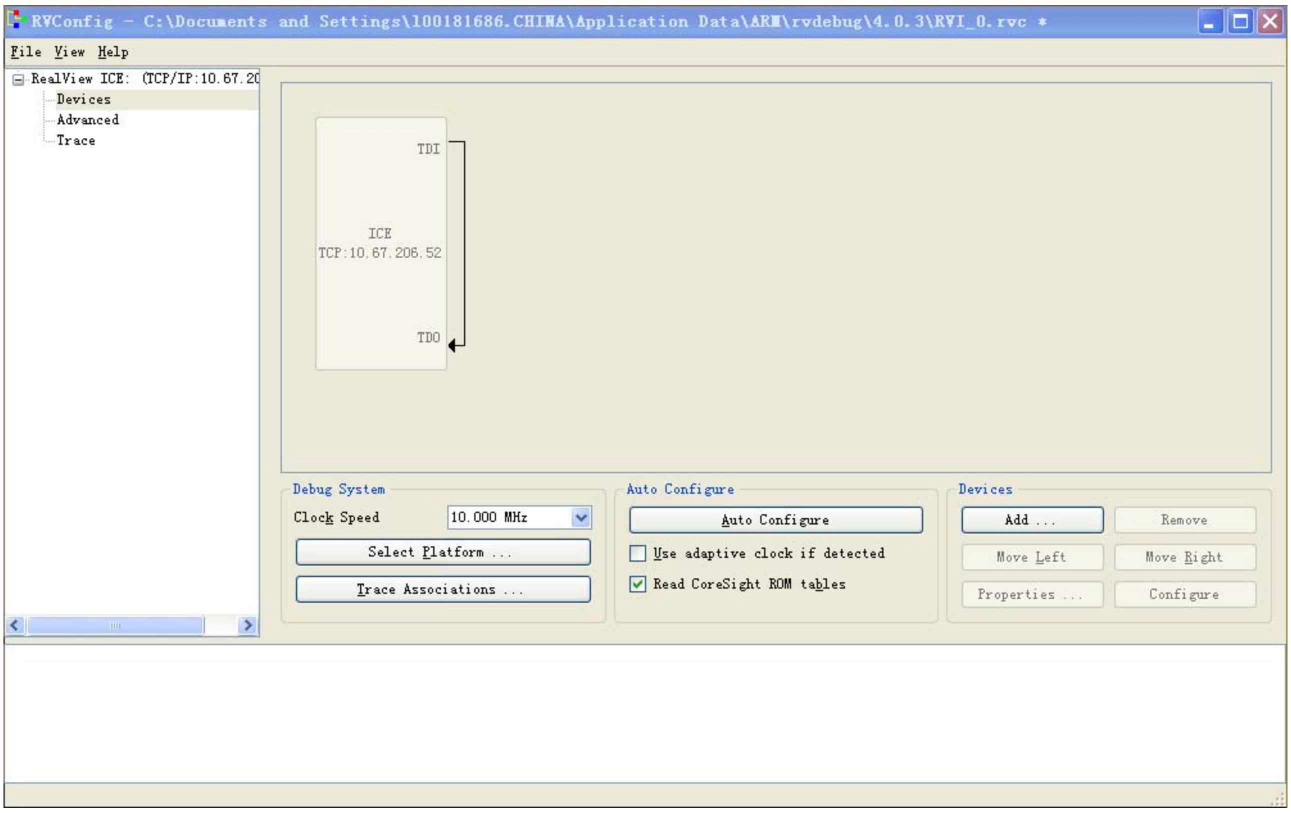
**Figure 4-3** Choosing the RealView-ICE


RVConfig - C:\Documents and Settings\100181686.CHINA\Application Data\ARM\rvdebug\4.0.3\RVI_0.rvc

```
Access   Host Name      IP Address      Ethernet Address
Found...1
⊟ TCP/IP
          bvtsoc        10.67.206.52    00:02:F7:00:3D:E8
```

**Step 5** Click **Auto Configure** shown in Figure 4-4. If the A9 processor is detected, the window shown in Figure 4-5 is displayed.

**Figure 4-4** Information box 1 of the RealView Debugger


RVConfig - C:\Documents and Settings\100181686.CHINA\Application Data\ARM\rvdebug\4.0.3\RVI_0.rvc *

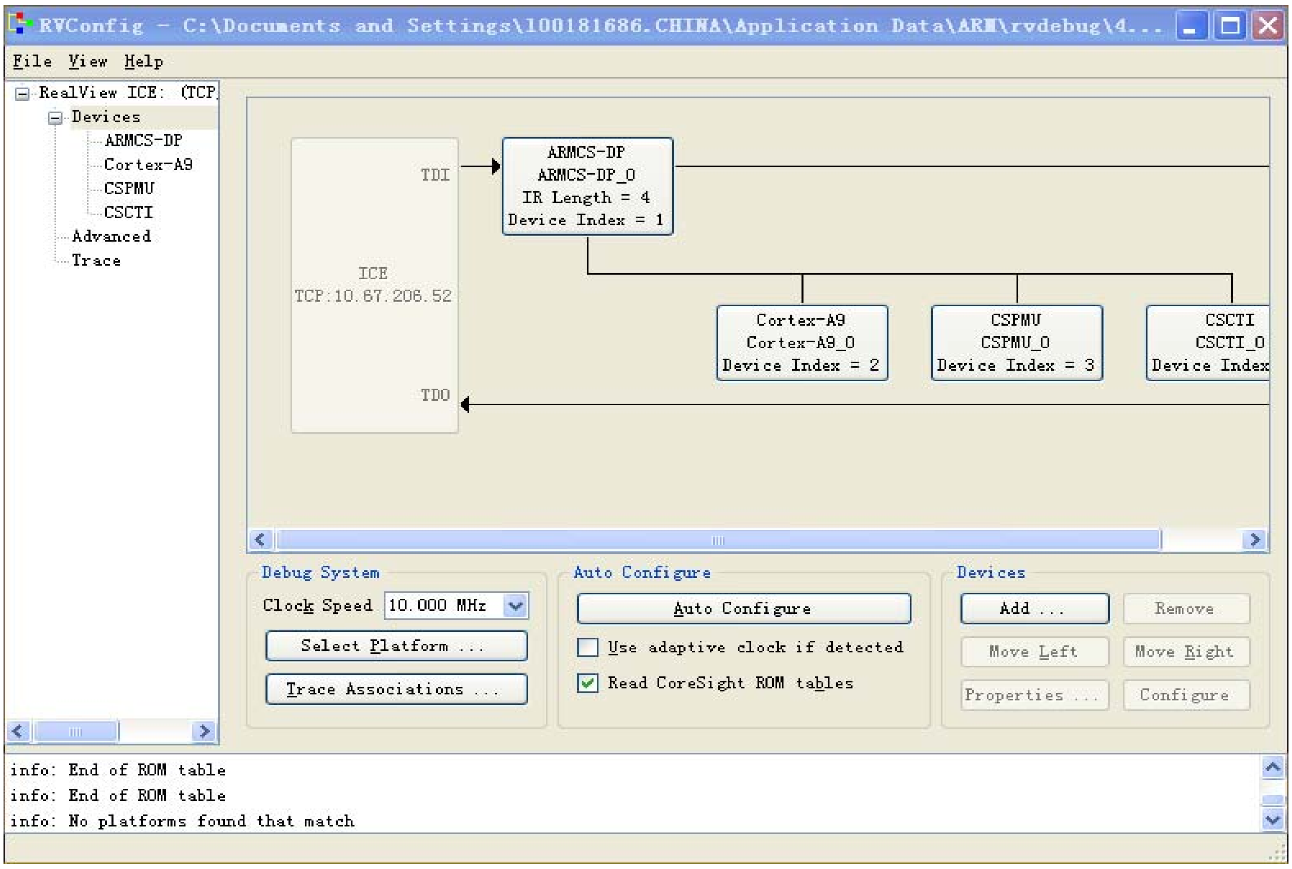
RealView ICE: (TCP/IP:10.67.20

ICE
TCP:10.67.206.52

Debug System
Clock Speed 10.000 MHz

Auto Configure
[ ] Use adaptive clock if detected
[✓] Read CoreSight ROM tables

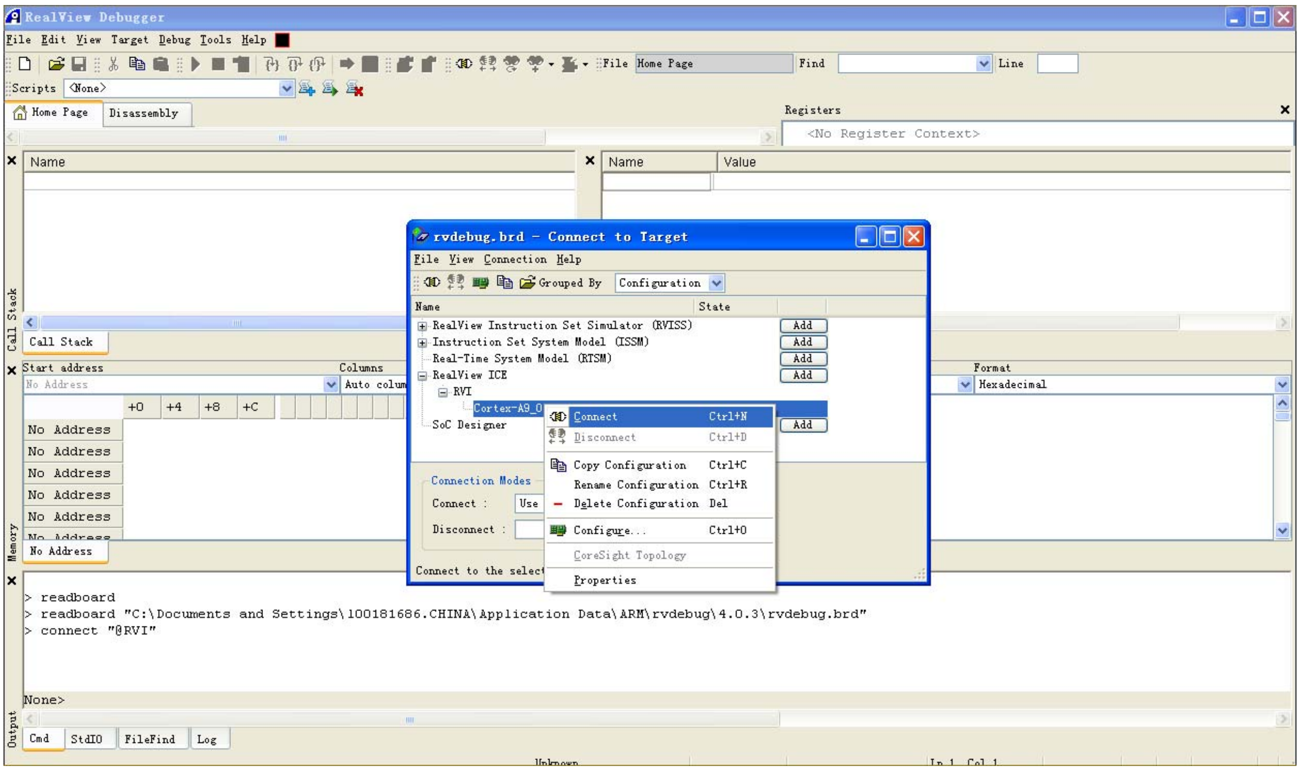
**Figure 4-5** Information box 2 of the RealView Debugger



```
info: End of ROM table
info: End of ROM table
info: No platforms found that match
```

**Step 6** Choose **File** > **Save** to save settings, click **Cortex-A9_0**, and choose **Connect** from the shortcut menu, as shown in Figure 4-6.

**Figure 4-6** Connect to A9


RealView Debugger
rvdebug.brd - Connect to Target
RealView Instruction Set Simulator (RVISS)
Instruction Set System Model (ISSM)
Real-Time System Model (RTSM)
RealView ICE
RVI
Cortex-A9_0
SoC Designer
Connect Ctrl+N
Disconnect Ctrl+D
Copy Configuration Ctrl+C
Rename Configuration Ctrl+R
Delete Configuration Del
Configure... Ctrl+O
CoreSight Topology
Properties

```
> readboard
> readboard "C:\Documents and Settings\100181686.CHINA\Application Data\ARM\rvdebug\4.0.3\rvdebug.brd"
> connect "@RVI"
```

**----End**