Hi3518A/Hi3518C/Hi3518E/Hi3516C U-boot Porting

Development Guide

Issue 01

Date 2014-02-26

HiSilicon Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 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 Hi3518A/Hi3518C/Hi3518E/Hi3516C board) on the Hi3518A/Hi3518C/Hi3518E/Hi3516C board, and how to using ARM debugging tools.

Related Version

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

Product Name	Version
Hi3518A	V100
Hi3518C	V100
Hi3518E	V100
Hi3516C	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 01 (2014-02-26)

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

Hi3518E information is added.

Issue 00B30 (2012-12-26)

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

Hi3516C information is added.

Issue 00B20 (2012-11-25)

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

Chapter 3 Burning the U-boot

In section 3.2, the initialization script name is changed.

Issue 00B01 (2012-08-30)

This issue is the first draft release.

Contents

Figures

Tables

1 Overview

NOTE

The principles and operations for the Hi3518A, Hi3518C, Hi3518E and Hi3516C are almost the same. The following uses the Hi3518A as an example to describe chip operations.

1.1 Description

The bootloader of the Hi3518A 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, see 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.
cpu	Indicates the code of CPUs and the entry code of the U-boot.
board	Indicates the code of boards, such as the memory driver code.
board/hi3518/	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.

Directory	Description
arch	Indicates the code of the chip architecture and the entry code of the U-boot.
common	Indicates the implementation files of functions or commands.
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 Hi3518A board:

- Double-data rate (DDR) synchronous dynamic random access memory (SDRAM): K4B1G1646E-HCH9
- NAND flash: TC58NVG1S3ETA00
- Serial peripheral interface (SPI) flash: MX25L12835E

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 Modifying the Flash Driver

The U-boot supports all the flash memories listed in the compatible table in **00.hardware/board/documents_cn**. In this case, no porting is required. If other flash memories are used, special porting is required.

2.3 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-hisiv100nptl-linux- hi3518a_config
//Select the corresponding board
make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux-
```

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

NOTE

- Use the **hi3518c_config** file to compile the Hi3518C U-boot, the **hi3518e_config** file to compile the Hi3518E U-boot and the **hi3516c_config** file to compile the Hi3516C U-boot.
- **CROSS_COMPILE=arm-hisiv100nptl-linux-** specifies that the compilation tool chain is arm-hisiv100nptl-linux-. If you want to use the arm-hisiv200-linux- tool chain, set the **CROSS_COMPILE** parameter to **CROSS_COMPILE=arm-hisiv200-linux-**.
- The **u-boot.bin** file is not the final U-boot image.

2.4 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.

2.5 Configuring Pin Multiplexing

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

2.6 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/tools/pc_tools/uboot_tools** of the SDK, and run the following command:

```
mkboot.sh reg_info.bin u-boot-200MHZ.bin
```

u-boot-200MHZ.bin is the final 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 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 Hi3518A SDK. When different DDRs are used, the Hi3518A can work properly only when the initialization scripts are reconfigured.

3.2 Initialization Scripts and U-boot

Similar to the U-boot, the initialization scripts are used to initialize the memory controller. The **u-boot-200MHZ.bin** file is stored in **package/images** of the SDK. In **osdrv/tools/pc_tools/uboot_tools**, the following two initialization scripts are provided:

- V1.0_Hi3518AV100_MDDRC_ASIC_1to2_440MHz_200MHz_400MHz_SiglCH_EccOff_ExOn_init_script.log
- V1.0_Hi3518CV100_MDDRC_ASIC_1to2_440MHz_200MHz_400MHz_SiglCH_EccOff_ExOn_init_script.log
- V1.0_Hi3518EV100_MDDRC_ASIC_1to2_440MHz_200MHz_400MHz_SiglCH_EccOff_ExOn_init_script.log
- V1.0_Hi3516CV100_MDDRC_ASIC_1to2_440MHz_220MHz_440MHz_SiglCH_EccOff_ExOn_init_script.log

NOTE

The three initialization files in **osdrv/tools/pc_tools/uboot_tools** apply to the Hi3518A, Hi3518C, Hi3518E and Hi3516C respectively.

3.3 Burning the U-boot by Using the RealView Debugger4.0

To burn the U-boot by using the RealView Debugger4.0, do as follows in sequence:

- Starting the RealView Debugger4.0
- Loading and Running the Initialization Script
- Opening the Image

3.3.1 Starting the RealView Debugger4.0

NOTE

For details about how to use the RealView Debugger4.0, see Chapter 4 "Using ARM Debugging Tools."

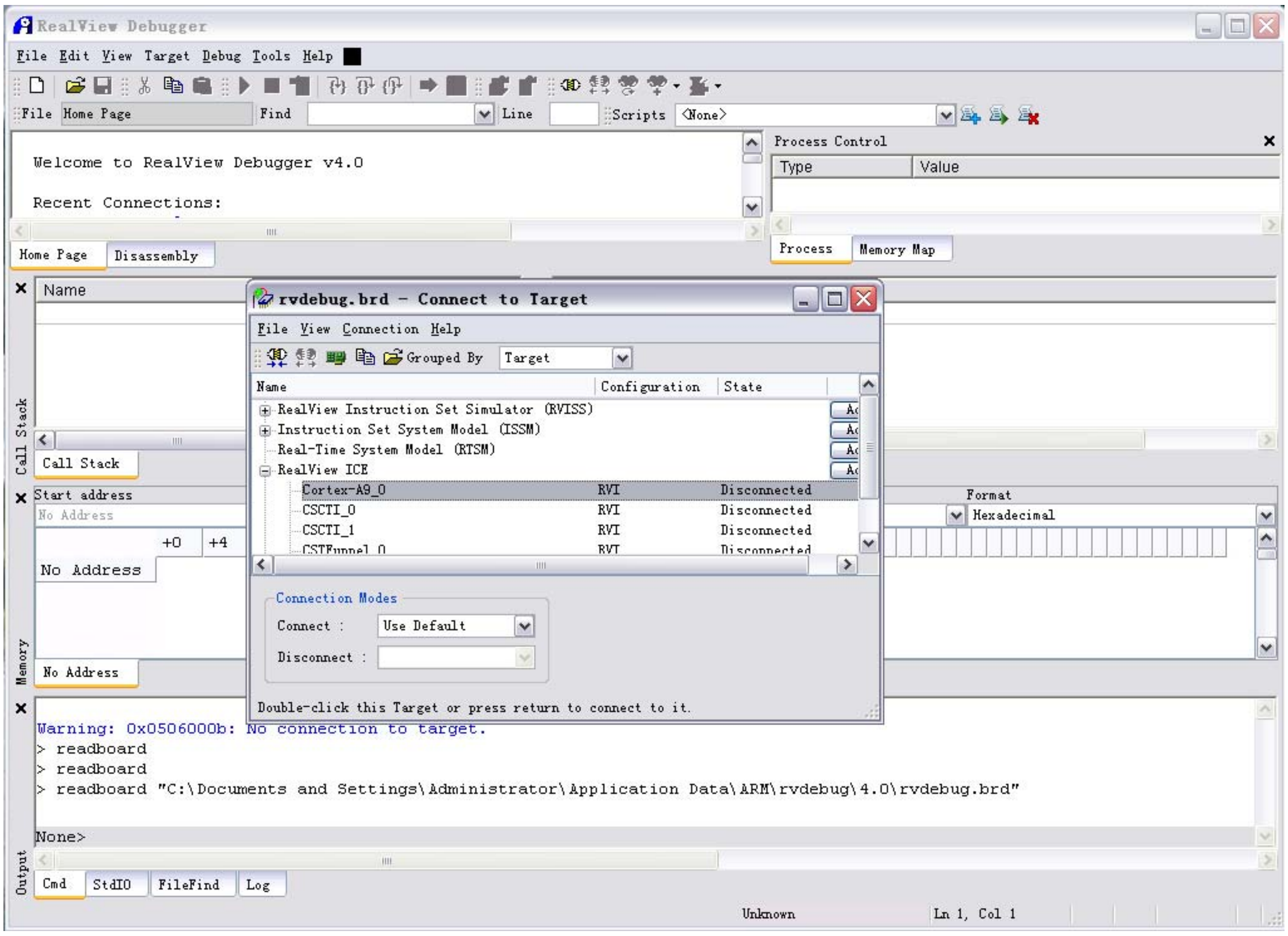
Perform the following steps:

Step 1 Run the RealView Debugger4.0.

Step 2 Locate the RealView ICE simulator.

Step 3 Connect the RealView Debugger4.0 to the ARM9 processor on the target board, see Figure 3-1.

Figure 3-1 RealView Debugger



```
Welcome to RealView Debugger v4.0

Recent Connections:
```

Name	Configuration	State
RealView Instruction Set Simulator (RVISS)		
Instruction Set System Model (ISSM)		
Real-Time System Model (RTSM)		
RealView ICE		
Cortex-A9_0	RVI	Disconnected
CSCTI_0	RVI	Disconnected
CSCTI_1	RVI	Disconnected
CSTFunnel_0	RVI	Disconnected

Double-click this Target or press return to connect to it.

```
Warning: 0x0506000b: No connection to target.
> readboard
> readboard
> readboard "C:\Documents and Settings\Administrator\Application Data\ARM\rvdebug\4.0\rvdebug.brd"

None>
```

Step 4 Double-click **ARM926EJ-S_0** to connect to the board.

----End

3.3.2 Loading and Running the Initialization Script

Perform the following steps:

Step 1 Choose **Tools** > **Include Commands from File**.

Step 2 In the **Include Commands from File** dialog box, select **V1.0_Hi3518AV100_MDDRC_ASIC_1to2_440MHz_200MHz_400MHz_SiglCH_EccOff_ExOn_init_script.log**, and click **Open**. The initialization script is loaded to the memory.

Step 3 Click ▶ shown in Figure 3-1 or press **F5**. Then the initialization script runs.

----End

3.3.3 Opening the Image

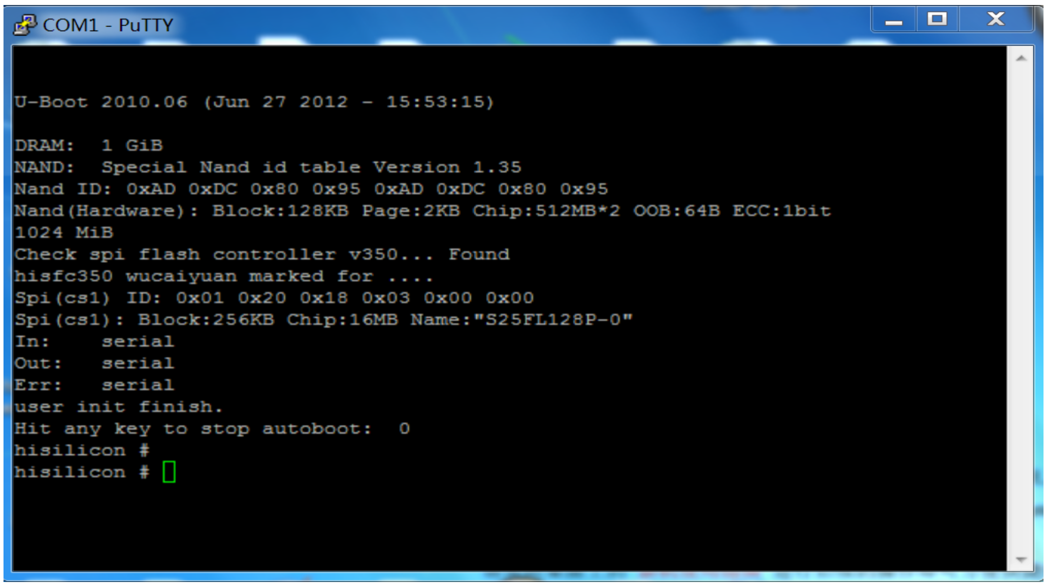
Perform the following steps:

Step 1 Choose **Target** > **Load image**, as shown in Figure 3-1.

Step 2 In the **Load Image** dialog box, select **u-boot**, and click **Open**. The **u-boot** image is loaded to the memory.

Step 3 Click ▶ or press **F5**. The information shown in Figure 3-1 is displayed in the serial port terminal window. In this case, the U-boot runs in the DDR

Figure 3-2 Serial port terminal command window after u-boot is loaded



```
U-Boot 2010.06 (Jun 27 2012 - 15:53:15)

DRAM:  1 GiB
NAND:  Special Nand id table Version 1.35
Nand ID: 0xAD 0xDC 0x80 0x95 0xAD 0xDC 0x80 0x95
Nand(Hardware): Block:128KB Page:2KB Chip:512MB*2 OOB:64B ECC:1bit
1024 MiB
Check spi flash controller v350... Found
hisfc350 wucaiyuan marked for ....
Spi(cs1) ID: 0x01 0x20 0x18 0x03 0x00 0x00
Spi(cs1): Block:256KB Chip:16MB Name:"S25FL128P-0"
In:    serial
Out:   serial
Err:   serial
user init finish.
Hit any key to stop autoboot:  0
hisilicon #
hisilicon #
```

Step 4 Download the U-boot to the flash memory based on the flash memory type according to the description in section 3.4 "Burning the U-boot to Two Types of Flash Memories."

----End

After the U-boot is burnt, close the RealView Debugger. If the board is reset or powered on, the U-boot starts automatically with the board.

3.4 Burning the U-boot to Two Types of Flash Memories

3.4.1 Burning the U-boot to the SPI Flash

Perform the following steps:

Step 1 Run the following commands in the serial port terminal after the U-boot runs in the memory:

```
hisilicon# mw.b 0x82000000 ff 0x100000     /*Set the DDR value to all
Fs.*/
hisilicon# tftp 0x82000000 u-boot-200MHZ.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

3.4.2 Burning the U-boot to the NAND Flash

Perform the following steps:

Step 1 Run the following commands in the serial port terminal 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-200MHZ.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

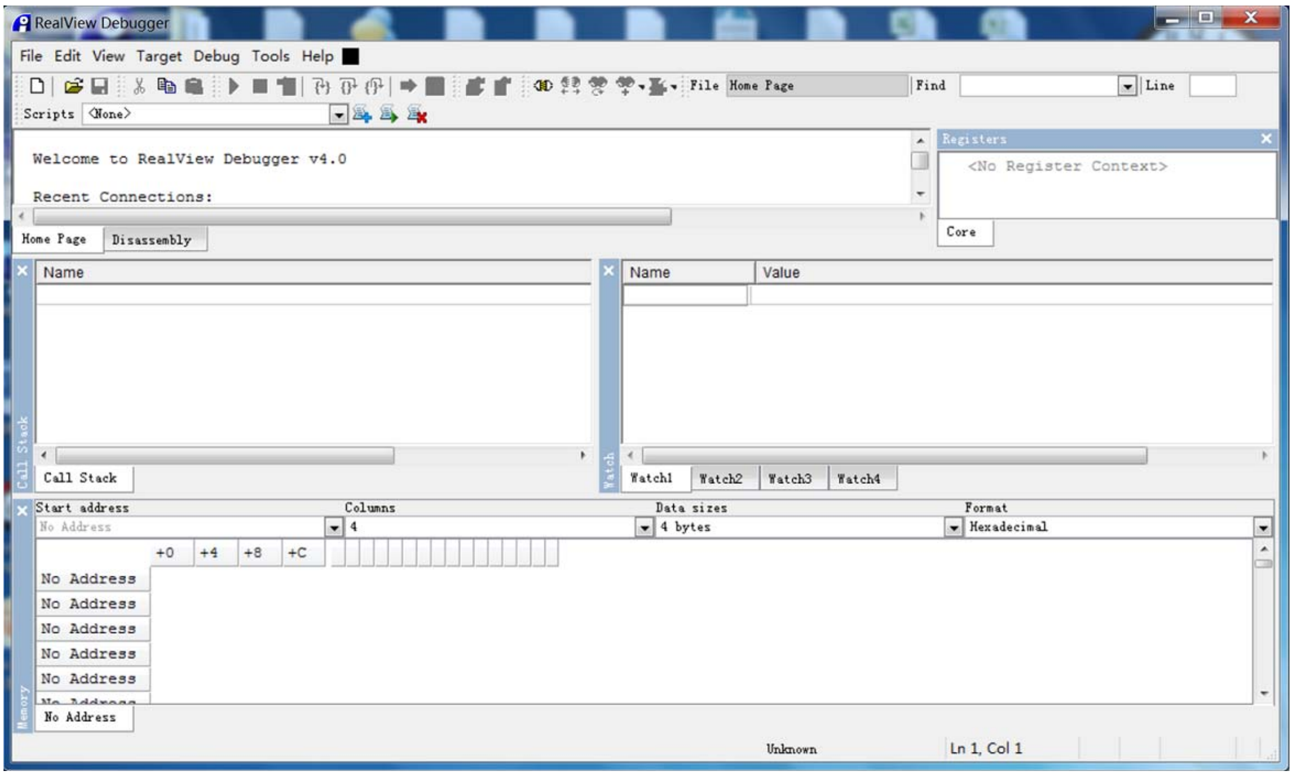
Before debugging programs or burning the U-boot to the board by using the Multi-ICE server, you must run the Multi-ICE server to locate the ARM chip connected to the hardware, and start the RealView Debugger.

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



Welcome to RealView Debugger v4.0

Recent Connections:

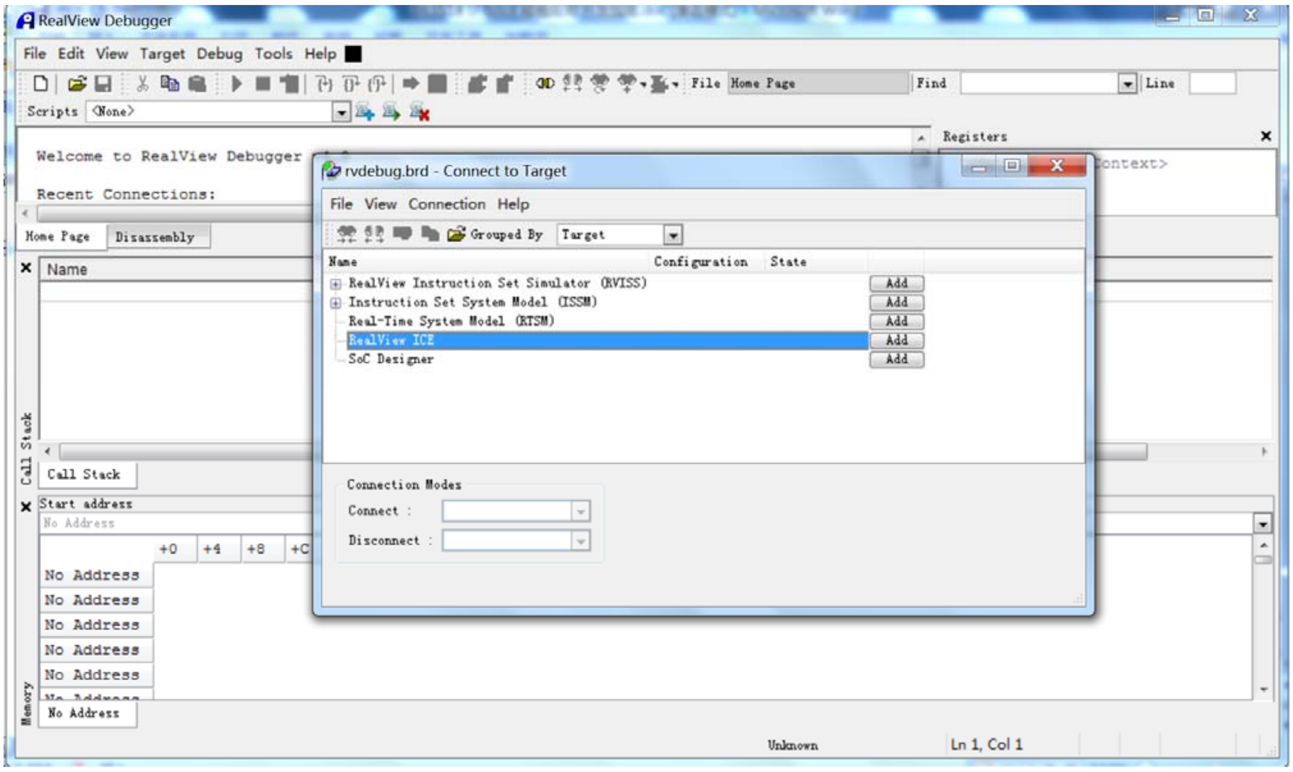
<No Register Context>

Start address: No Address; Columns: 4; Data sizes: 4 bytes; Format: Hexadecimal

No Address
No Address
No Address
No Address
No Address
No Address

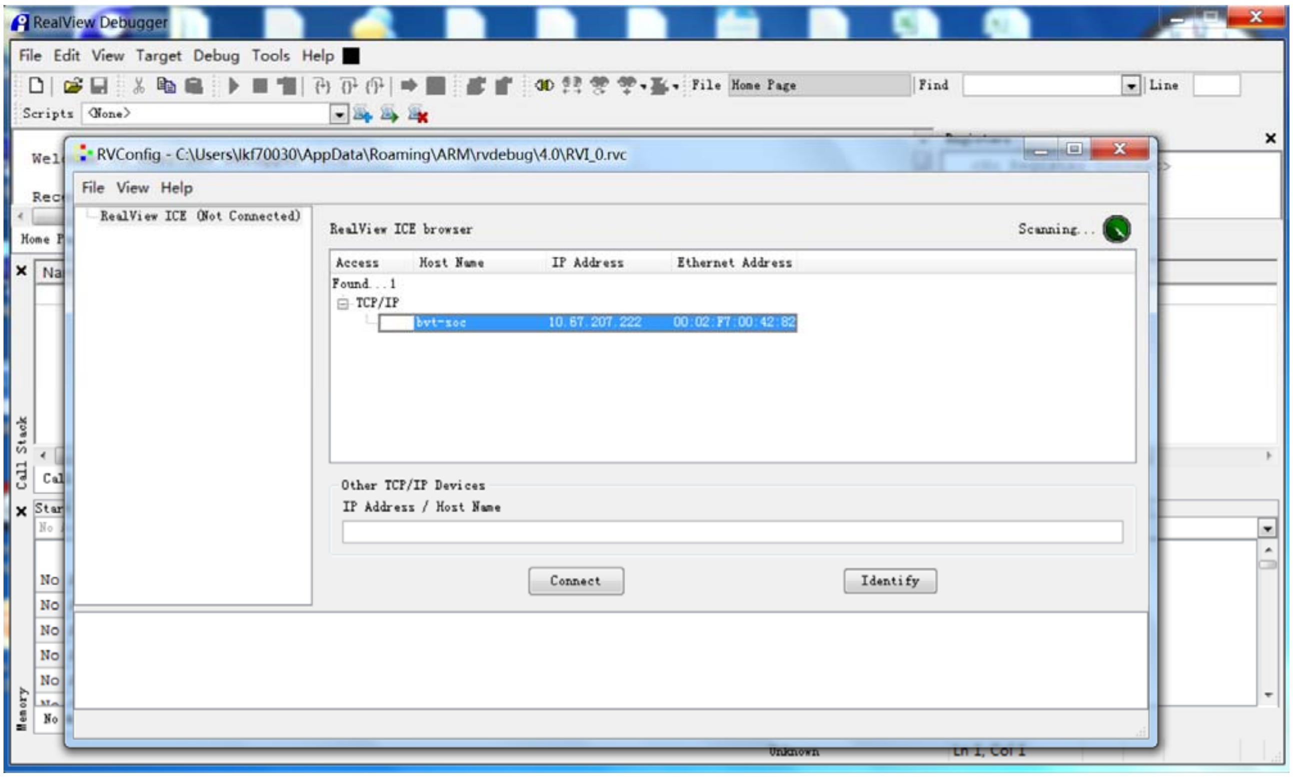
Step 3 Click to connect the simulator to the RealView-ICE. The window shown in Figure 4-2 is displayed.

Figure 4-2 Connect to Target window



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


RealView Debugger
File Edit View Target Debug Tools Help
RVConfig - C:\Users\lkf70030\AppData\Roaming\ARM\rvdebug\4.0\RVI_0.rvc
File View Help
RealView ICE (Not Connected)
RealView ICE browser
Scanning...
Other TCP/IP Devices
IP Address / Host Name
Connect
Identify

Access	Host Name	IP Address	Ethernet Address
Found...1			
TCP/IP			
	bvt-soc	10.67.207.222	00:02:F7:00:42:82

Step 5 Click **Auto Configure** shown in Figure 4-4. If the ARM9 processor is detected, the window shown in Figure 4-5 is displayed.

Figure 4-4 Information box 1 of the RealView Debugger

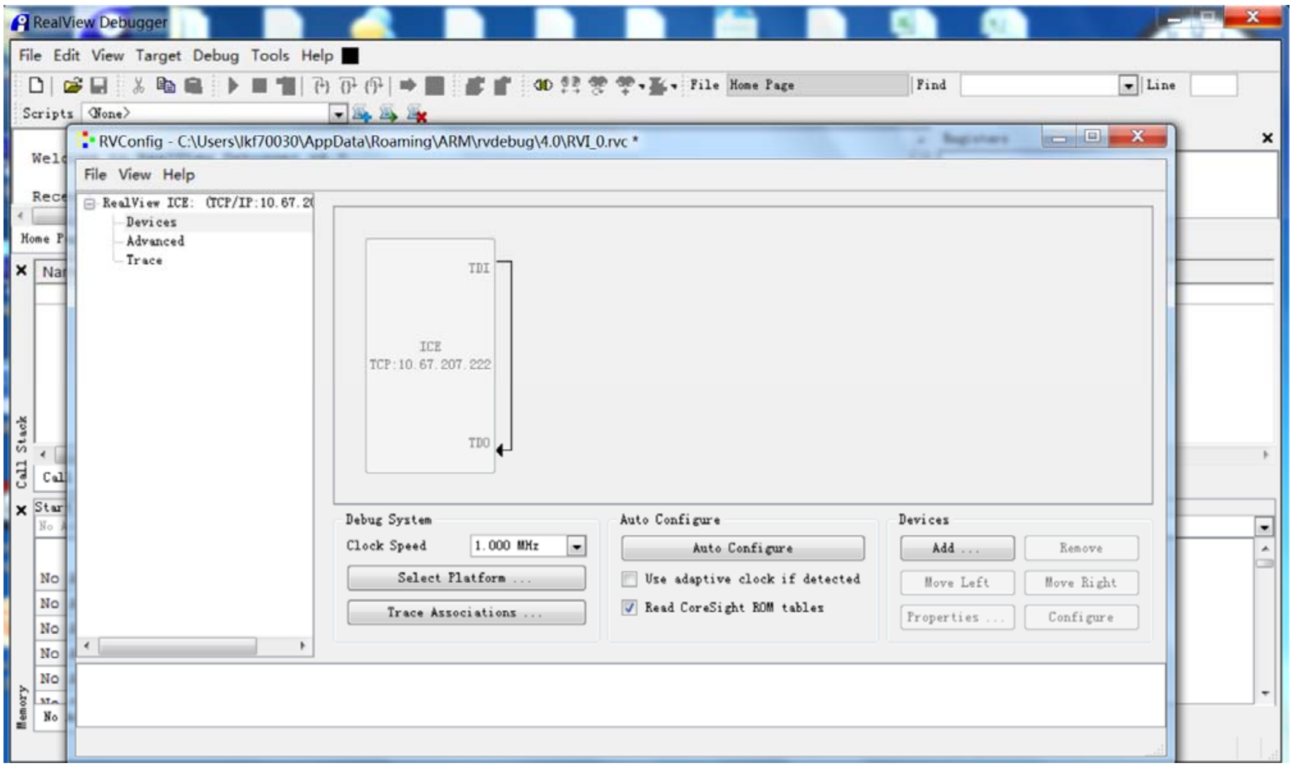
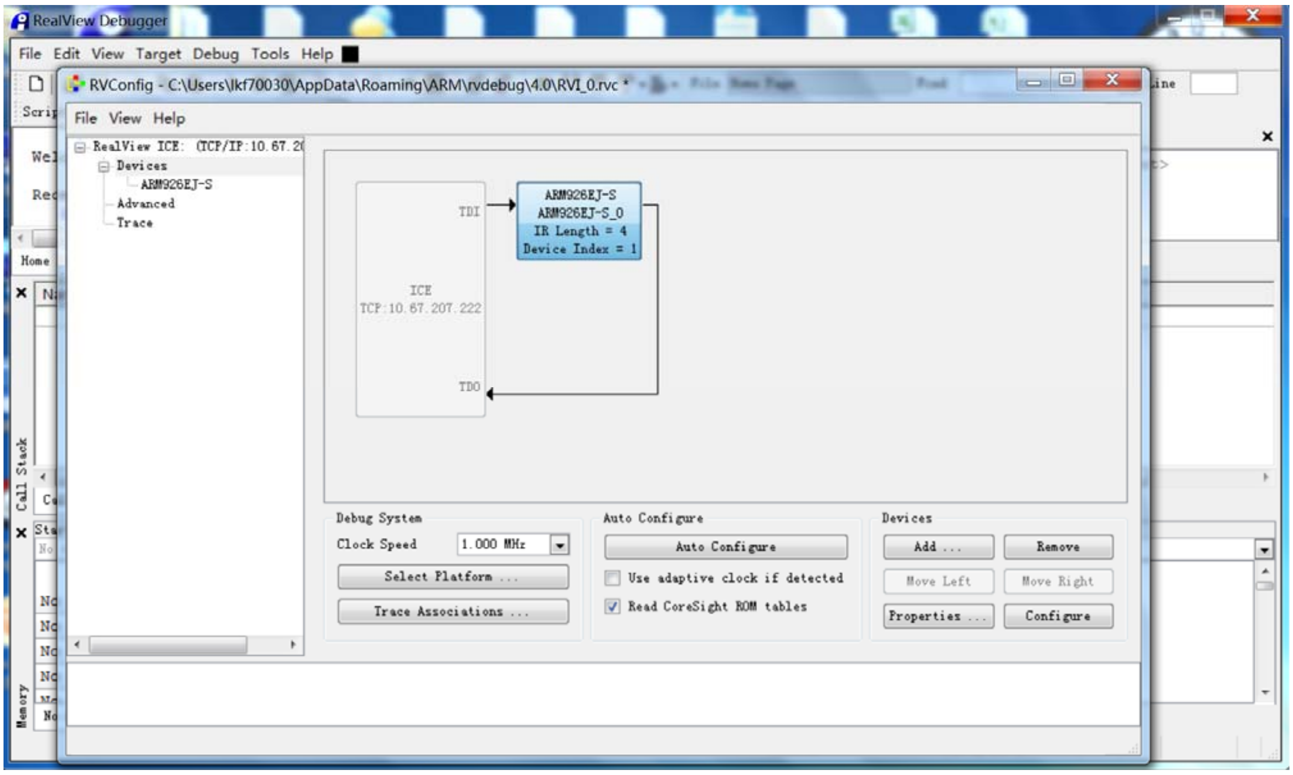
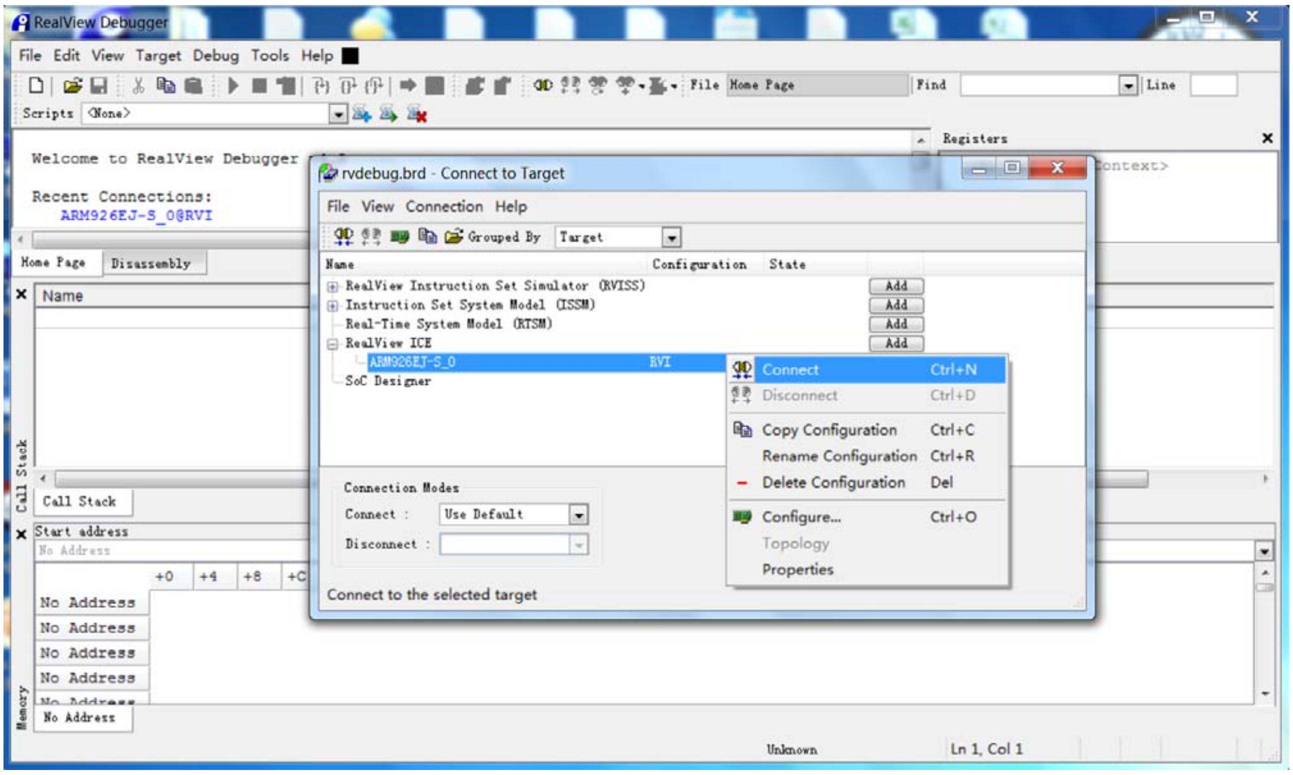


Figure 4-5 Information box 2 of the RealView Debugger



Step 6 Choose **File** > **Save** to save settings, click **ARM926EJ-S_0**, and choose **Connect** from the shortcut menu, as shown in Figure 4-6.

Figure 4-6 Connect to ARM9


RealView Debugger
File Edit View Target Debug Tools Help
rvdebug.brd - Connect to Target
File View Connection Help
Grouped By Target
Name Configuration State
RealView Instruction Set Simulator (RVISS)
Instruction Set System Model (ISSM)
Real-Time System Model (RTSM)
RealView ICE
ARM926EJ-S_0 RVI
SoC Designer
Connect Ctrl+N
Disconnect Ctrl+D
Copy Configuration Ctrl+C
Rename Configuration Ctrl+R
Delete Configuration Del
Configure... Ctrl+O
Topology
Properties
Connection Modes
Connect : Use Default
Disconnect :
Connect to the selected target

----End