H-JTAG
用户使用手册

WWW.HJTAG.COM

H-JTAG

用户使用手册

修改记录

日期	版本	改动
2007-10-01	A	发布第一版本
2007-11-30	B	修改版本
2008-03-03	C	更正了 TAP 的设置说明
2009-01-08	D	软件更新

版权声明

1. 文档中出现的 JTAG 表述，为 IEEE-1149 标准，其所有权属于国际电子电气协会；
2. 文档中出现的所有 ARM 标识和表述，均为 ARM 公司的注册商标，其所有权属于 ARM 公司；
3. 文档中提及的任何第三方的注册商标和产品标识，均属于第三方公司所有；
4. 如果文档当中有任何地方侵犯了您的权利和版权，请和我们联系，我们将及时修改；
5. 本文档为开放文档，用户可以在保证文档完整性的前提下，自由分发；

官方主页

HTTP://WWW.HJTAG.COM

技术支持论坛

HTTP://FORUM.HJTAG.COM

目 录

前 言

A. 关于本手册

H-JTAG 用户使用手册简单介绍了 H-JTAG 和 H-FLASHER 的基本使用和配置，手册同时也提供了一些简单的例子供用户参考。如果用户需要更多的相关信息，请访问 H-JTAG 主页 WWW.HJTAG.COM 或技术支持论坛 HTTP://FORUM.HJTAG.COM。

B. 适合的读者

如果你是一个初级用户，打算使用 H-JTAG 进行调试和开发，本手册可作为一个快速入门指南。如果你是高级用户，本手册可以作为参考，用户可以选择性的阅读某些章节。

C. 意见反馈

如果你发现本手册中有不正确的地方，或者有什么好的建议，请发送电子邮件至 twentyone@hjtag.com。

第一章 H-JTAG 介绍

1.1 H-JTAG 介绍

H-JTAG 是一款简单易用的的调试代理软件，功能和流行的 MULTI-ICE 类似。H-JTAG 包括三个工具软件：H-JTAG SERVER，H-FLASHER 和 H-CONVERTER。其中，H-JTAG SERVER 实现调试代理的功能，H-FLASHER 实现了 FLASH 烧写的功能，H-CONVERTER 是一个简单的文件格式转换工具，支持常见文件格式的转换。H-JTAG 的基本结构如下图所示。

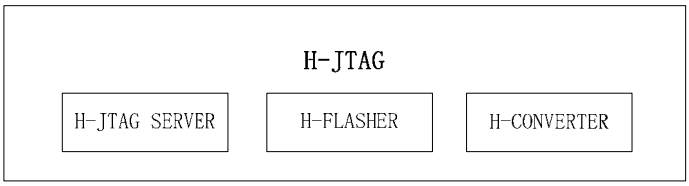

图 1-1 H-JTAG 软件结构

H-JTAG 支持所有基于 ARM7， ARM9，XSCALE 和 CORTEX-M3 芯片的调试，并且支持大多数主流的 ARM 调试软件，如 ADS、RVDS、IAR 和 KEIL。通过灵活的接口配置，H-JTAG 可以支持 WIGGLER，SDT-JTAG，用户自定义的各种 JTAG 调试小板和 H-JTAG USB 仿真器。同时，附带的 H-FLASHER 烧写软件还支持常用片内片外 FLASH 的烧写。使用 H-JTAG，用户能够方便的搭建一个简单易用的 ARM 调试开发平台。H-JTAG 的功能和特定总结如下：

1. 支持 RDI 1.5.0 以及 1.5.1；
2. 支持 所有 ARM7，ARM9，XSCALE 及 CORTEX-M3 芯片；
3. 支持 THUMB 以及 ARM 指令；
4. 支持 LITTLE-ENDIAN 以及 BIG-ENDIAN；
5. 支持 SEMIHOSTING；
6. 支持 WIGGLER, SDT-JTAG，自定义 JTAG 调试板和 H-JTAG USB 仿真器；
7. 支持 WINDOWS 9.X/NT/2000/XP；
8. 支持常用 FLASH 芯片的编程烧写；
9. 支持 LPC2000，AT91SAM，LUMINARY 和 STM32F 系列的片内 FLASH 自动下载；

1.2 H-JTAG 调试/烧写结构

H-JTAG 支持 ARM 公司的 RDI 接口。通过 RDI 接口，H-JTAG 能够支持大多数主流的 ARM 调试软件。调试的结构如图 1-2 所示。

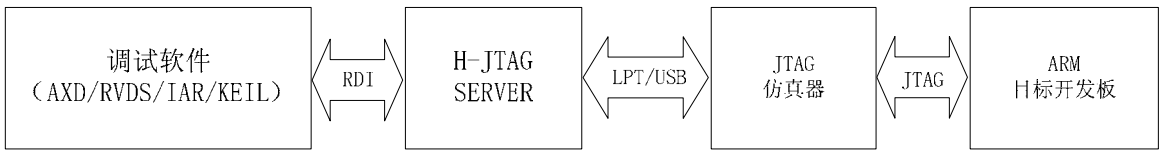

图 1-2 H-JTAG 调试结构

调试软件（AXD/RVDS/IAR/KEIL）通过 RDI 接口与 H-JTAG SERVER 进行交互。H-JTAG SERVER 通过并口/USB 连接 JTAG 仿真器。H-JTAG 提供了灵活的 JTAG 接口设置，同时支持并口及 USB 接口仿真器。通过设置，H-JTAG 可以支持不同类型的 JTAG 调试板，例如流行的 WIGGLER，SDT-JTAG，用户也可以根据自己的需要使用自定义的 JTAG 调试板。

除了调试，用户还可以通过 H-FLASHER 将程序和数据烧写/下载到 FLASH 芯片中去。目前，H-FLASHER 支持的常见的片内和片外 FLASH 芯片。随着软件的更新和升级，H-FLASHER 支持的芯片类型也会不断的增加。在执行 FLASH 烧写时，H-FLASHER 与 H-JTAG SERVER 的连接如下图所示：

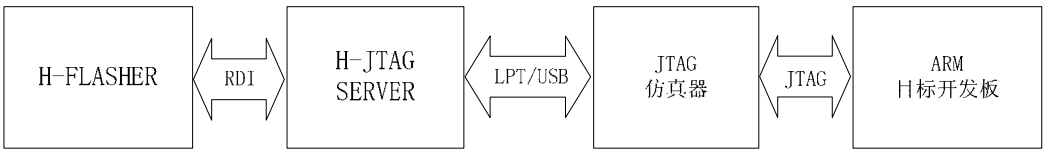

图 1-3 H-FLASHER 与 H-JTAG SERVER 的连接

烧写 FLASH 时，H-FLASHER 与 H-JTAG 的连接交互和调试的时候十分类似。H-FLASHER 通过 RDI 接口与 H-JTAG SERVER 进行交互，以访问和控制目标开发板。根据用户提供的配置文件，H-FLASHER 最终完成 FLASH 烧写工作。

第二章 H-JTAG 的安装和 GUI 介绍

本章将简单介绍 H-JTAG 的安装和卸载，以及 H-JTAG 与 H-FLASHER 的 GUI（图形用户接口）。具体的使用和设置，请参考 4-7 章节。

2.1 H-JTAG 的安装

用户可以从 H-JTAG 的主页 WWW.HJTAG.COM 下载最新版本的 H-JTAG 安装程序。一般情况下，用户下载的是一个压缩文件（RAR/ZIP）。解压后，可以得到 H-JTAG 的安装文件：H-JTAG.EXE。双击该文件，就可以开始进行安装。

首先，用户会看到一个欢迎页面，如图 2-1 所示。点击 NEXT，就可以进入到安装的下一步。

图 2-1 H-JTAG 安装步骤 1

在第二步，用户会看到 H-JTAG 的使用协议，如图 2-2 所示。请仔细阅读该协议。在同意后，请选择“I agree with the above terms and conditions”，表示接受。然后点击 NEXT，就可以进入到安装的下一步。

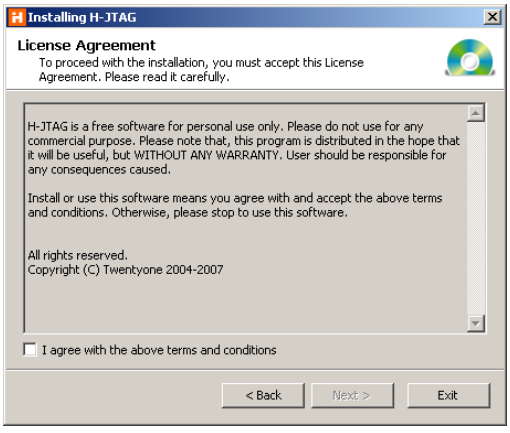

图 2-2 H-JTAG 安装步骤 2

在第三步，用户会看到如图 2-3 所示的页面。在该页面，用户可以选择 H-JTAG 的安装目录，或者使用默认的安装目录。选择好目录后，点击 NEXT 进入安装的下一步。

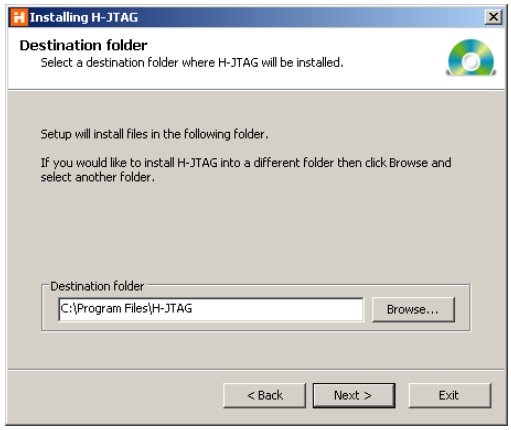

图 2-3 H-JTAG 安装步骤 3

在第四步当中，用户将会看到如图 2-4 所示的安装进度。该图表示安装正在进行，H-JTAG 将自动安装必要的文件和驱动程序。

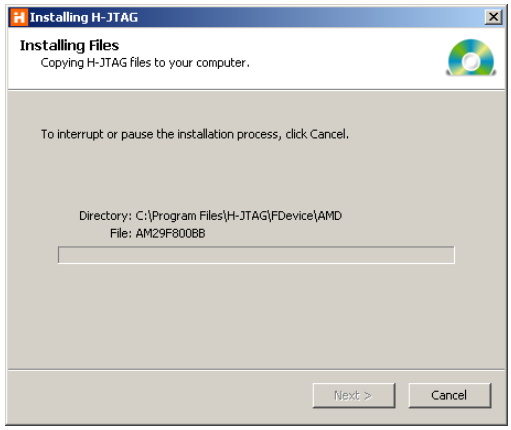

图 2-4 H-JTAG 安装步骤 4

安装完成后，用户将会看到如图 2-5 所示的提示，表示安装完成。按照提示，点击 FINISH 就可以完成最后的安装。

图 2-5 H-JTAG 安装步骤 5

安装完成后，H-JTAG 会在开始菜单和桌面上创建相应的快捷方式，如图 2-6 所示。

图 2-6 H-JTAG 快捷方式

2.2 H-JTAG 的卸载

如果用户要卸载 H-JTAG，请运行 H-JTAG 菜单下的 UNSTALL 程序。该程序将自动卸载 H-JTAG，并删除所有的相关文件。卸载过程当中，只需要按照提示操作即可。

2.3 H-JTAG 的 GUI

H-JTAG 运行时，用户主界面如图 2-7 所示。其中，（1）为 H-JTAG 菜单；（2）为 H-JTAG 工具栏；（3）为 H-JTAG 检测到的目标 CPU 的类型；（4）为目标 CPU 的芯片 ID；（5）依次显示了当前的调试软件，RDI 接口版本和硬件接口。

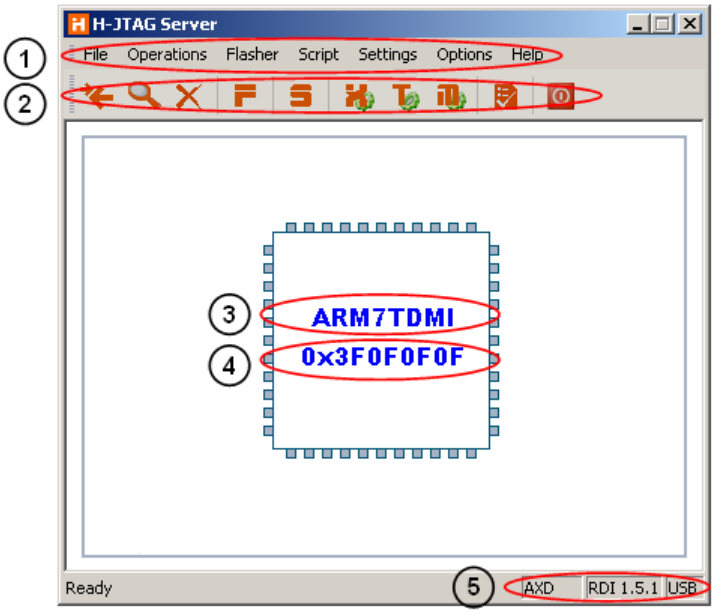

图 2-7 H-JTAG 主界面

H-JTAG 的菜单包括了 H-JTAG 的所有操作和设置，而工具栏则包括了大部分常用的操作和设置。当 H-JTAG 检测到连接的目标开发板后，在主界面的中央部分会显示芯片的类型和其 32 位芯片 ID。如果检测失败，或者芯片无法识别，H-JTAG 将会显示 UNKNOWN，提示用户 H-JTAG 无法检测/识别目标开放板。

2.3.1 H-JTAG 菜单介绍

- FILE 菜单，如图 2-8 所示：

图 2-8 H-JTAG FILE 菜单

  - EXIT – 退出 H-JTAG。

- OPERATIONS 菜单，如图 2-9 所示：

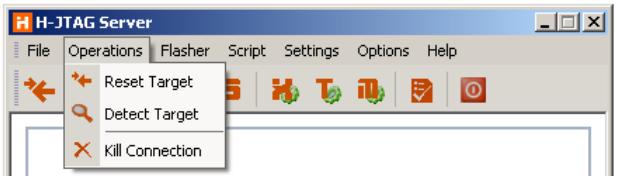

图 2-9 H-JTAG OPERATIONS 菜单

  - RESET TARGET – 复位调试目标
  - DETECT TARGET – 检测调试目标
  - KILL CONNECTION – 断开当前连接

- FLASHER 菜单，如图 2-10 所示：

图 2-10 H-JTAG FLASHER 菜单

  - START FLASHER – 启动 H-FLASHER
  - AUTO DOWNLOAD – 启用/禁用自动 FLASH 下载

- SCRIPT 菜单，如图 2-11 所示：

图 2-11 H-JTAG SCRIPT 菜单

  - INIT SCRIPT – 设置初始化脚本
  - AUTO INIT – 启用/禁用自动初始化

■ SETTINGS 菜单，如图 2-12 所示：

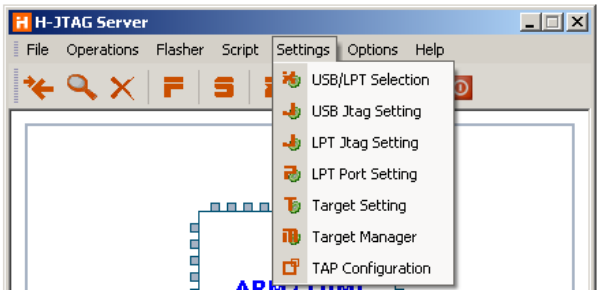

图 2-12 H-JTAG SETTINGS 菜单

- USB/LPT SELECTION – 并口/USB 接口选择
- USB JTAG SETTING – USB JTAG 设置
- LPT JTAG SETTING – LPT JTAG 设置
- LPT PORT SETTING – 并口设置
- TARGET SETTING – 调试目标设置
- TARGET MANAGER – 芯片 ID 管理器
- TAP CONFIGURATION – TAP 设置

■ OPTIONS 菜单，如图 2-13 所示：

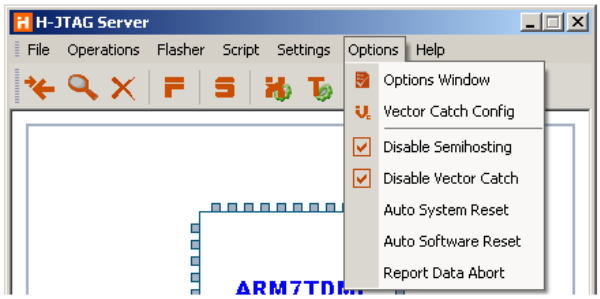

图 2-13 H-JTAG OPTIONS 菜单

- OPTIONS WINDOW – 打开选项设置主窗口
- VECTOR CATCH CONFIG – 中断捕获设置
- DISABLE SEMIHOSTING – 禁用 SEMIHOSTING 功能
- DISABLE VECTOR CATCH – 禁用中断向量捕获功能
- AUTO SYSTEM RESET – 启用/禁用自动系统复位
- AUTO SOFTWARE RESET – 启用/禁用自动软件复位
- REPORT DATA ABORT – 启用/禁用数据异常报告

■ HELP 菜单，如图 2-14 所示：

图 2-14 H-JTAG HELP 菜单

- H-JTAG HOME – 访问 H-JTAG 主页
- CHECK UPDATE – 检查更新
- ABOUT H-JTAG – 关于 H-JTAG

2.3.2 H-JTAG 工具栏介绍

H-JTAG 工具栏包括了菜单中使用频率比较高的大部分操作，方便用户使用。H-JTAG 的工具栏如下图所示。

图 2-15 H-JTAG 工具栏

工具栏上每个按钮的功能定义如下：

- 复位调试目标
- 检测调试目标
- 断开当前连接
- 启动 H-FLASHER
- 设置初始化脚本
- LPT/USB 接口选择
- 调试目标设置
- 芯片 ID 管理器
- 打开选项设置窗口
- 退出 H-JTAG

2.3.3 H-JTAG 系统托盘菜单

当 H-JTAG 最小化的时候，主窗口会自动隐藏来，只在系统托盘上显示 H-JTAG 的图标。左键单击该图标，可以恢复 H-JTAG 的主窗口。右键单击该图标，可以显示系统托盘菜单，其中包括了常用的一些操作和设置按钮。系统托盘菜单如图 2-16 所示：

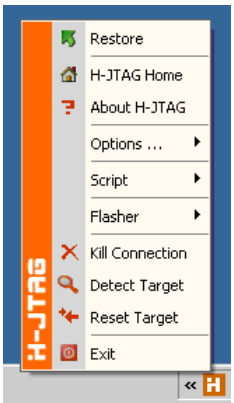

图 2-16 H-JTAG 系统托盘菜单

H-JTAG 系统托盘菜单的功能定义如下：

- RESTORE – 恢复主窗口
- H-JTAG HOME – 访问主页

- ABOUT H-JTAG – 关于 H-JTAG
- OPTIONS – 选项菜单
- SCRIPT – 脚本菜单
- FLASHER – H-FLASHER 菜单
- KILL CONNECTION – 断开当前连接
- DETECT TARGET – 检测调试目标
- RESET TARGET – 复位调试目标
- EXIT – 退出 H-JTAG

2.4 H-FLASHER 的 GUI

H-FLASHER 运行时，其主窗口如图 2-17 所示。其中（1）为菜单；（2）为 FLASH 编程向导区域；（3）为 FLASH 烧写/配置区域。在 FLASH 编程向导当中，可以选择不同的步骤。根据当前向导中的选择，配置区域会显示不同的设置。具体请参考第五章。

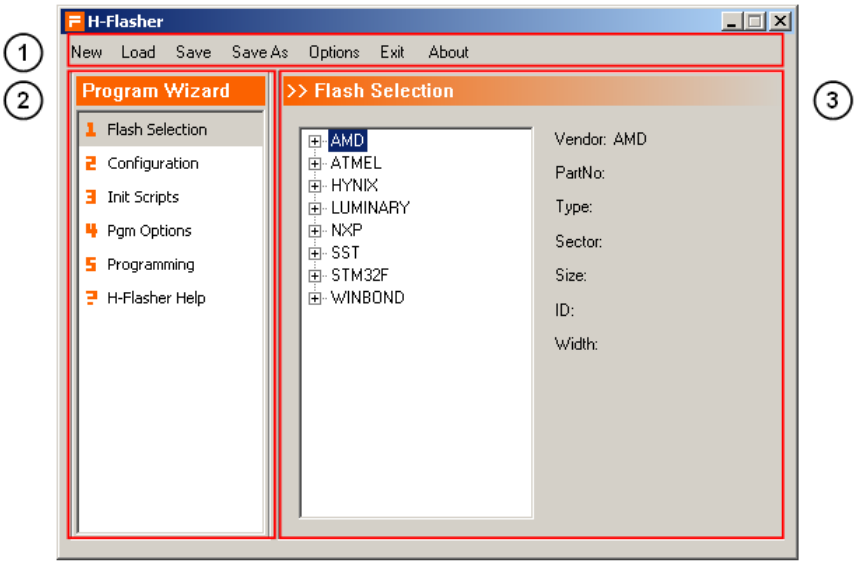

图 2-17 H-FLASHER 主界面

2.4.1 H-FLASHER 的菜单介绍

H-FLASHER 的主菜单如图 2-18 所示：

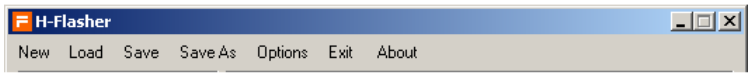

图 2-18 H-FLASHER 主菜单

H-FLASHER 菜单的具体定义如下：

- NEW – 创建新的配置文件
- LOAD – 装载配置文件
- SAVE – 保存当前配置文件
- SAVEAS – 将当前的配置文件另存为
- OPTIONS – 设置选项

- EXIT – 退出 H-FLASHER
- ABOUT – 关于 H-FLASHER

2.4.2 H-FLASHER 的系统托盘菜单

当 H-FLASHER 最小化的时候，主窗口会自动隐藏来，只在系统托盘上显示 H-FLASHER 的图标。左键单击该图标，可以恢复 H-FLASHER 的主窗口。右键单击该图标，可以显示系统托盘菜单，如图 2-19 所示：

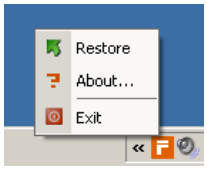

图 2-19 H-FLASHER 系统托盘菜单

H-FLASHER 系统托盘菜单的具体定义如下：

- RESTORE – 恢复主窗口
- ABOUT – 关于 H-FLASHER
- EXIT – 退出 H-FLASHER

2.4.3 H-FLASHER 的编程向导

这个章节将简单介绍 H-FLASHER 的编程向导，具体的设置请参考后面的章节。编程向导总共包括五个步骤与一个帮助部分。下面我们一起看看每个步骤的设置。

（1）Flash Selection

Flash Selection 是编程向导中的第一步，如下图所示。在这一步，用户需要确定烧写的目标芯片是什么。H-FLASHER 支持的的所有 FLASH 芯片都按照生产厂商分类列表。当用户选定一块片子的时候，右边会显示该芯片的基本信息，例如芯片类型，容量和 ID 等。

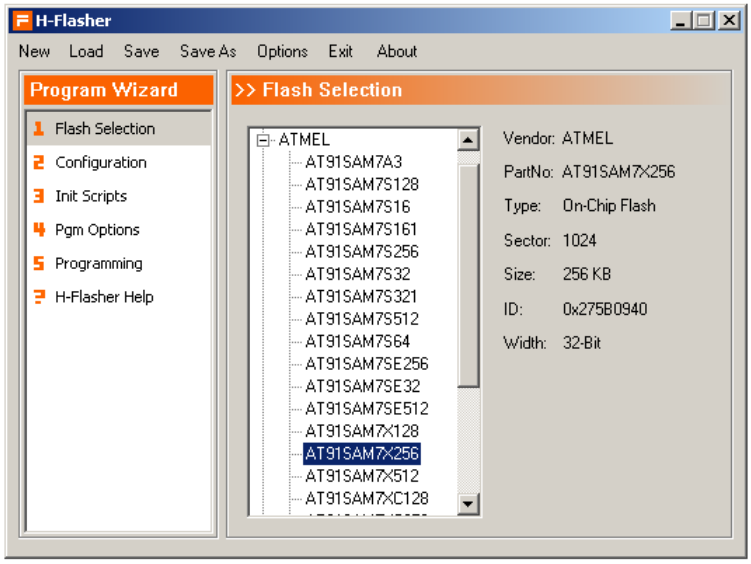

图 2-20 H-FLASHER 编程向导－Flash Selection

（2）Configuration

Configuration 是编程向导中的第二步，如下图所示。在这一步，用户需要设置基本的信息，例如 FLASH 芯片的位宽，FLASH 的起始地址，RAM/SDRAM 的起始地址，外部晶振的频率，初始化 TCK 和编程 TCK 等。这些都是必须的信息，H-FLASHER 会根据这些信息来完成 FLASH 烧写。如果这个步骤中的设置框为灰色，说明对于当前选择的 FLASH 芯片来说，这些信息都是固定的，不需要用户提供。对大部分的片内 FLASH 而言，这些信息都是固定的，基本不需要用户提供。

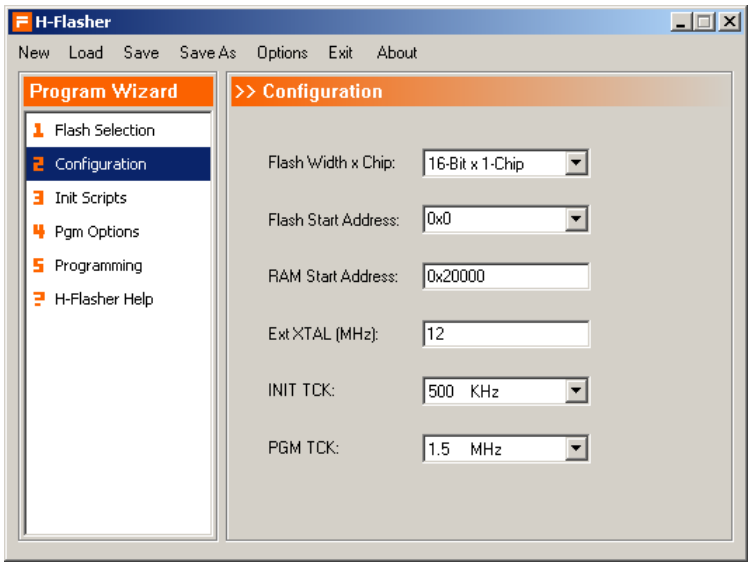

图 2-21 H-FLASHER 编程向导－Configuration

（3） Init Script

Init Script 是编程向导中的第三步，如下图所示。在这个步骤当中，用户可以输入必要的初始化脚本，用以配置目标系统。在烧写的时候，H-FLASHER 首先会执行用户提供的初始化脚本，以初始化目标系统。然后在根据用户提供的存储信息，对目标 FLASH 进行操作。对于片内 FLASH 而言，H-FLASHER 的驱动当中已经包括了初始化，所以用户不需要提供初始化脚本。此时，脚本编辑功能会自动被禁用掉。

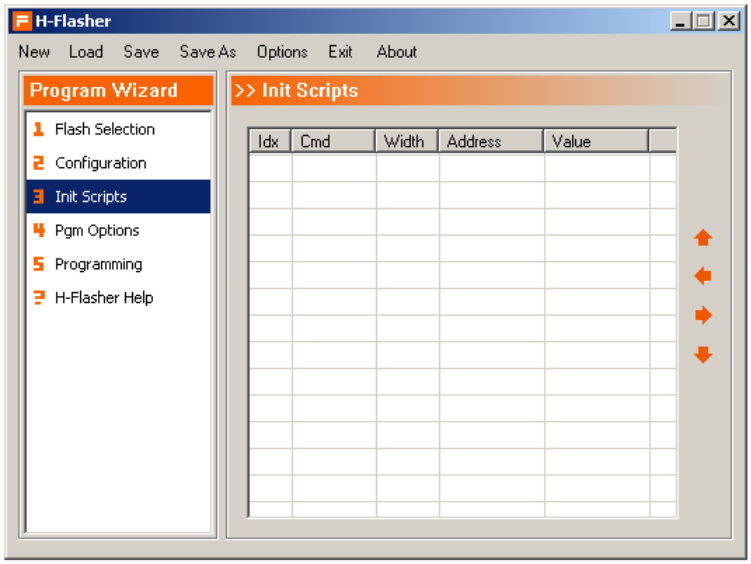

图 2-22 H-FLASHER 编程向导－Init Script

（4） Program Options

Program Options 编程向导中的第四步，如下图所示。在这个步骤中，用户可以根据选择的 FLASH 芯片和自己的需要选择不通的选项。常用的选项包括复位，二次验证和加密等。

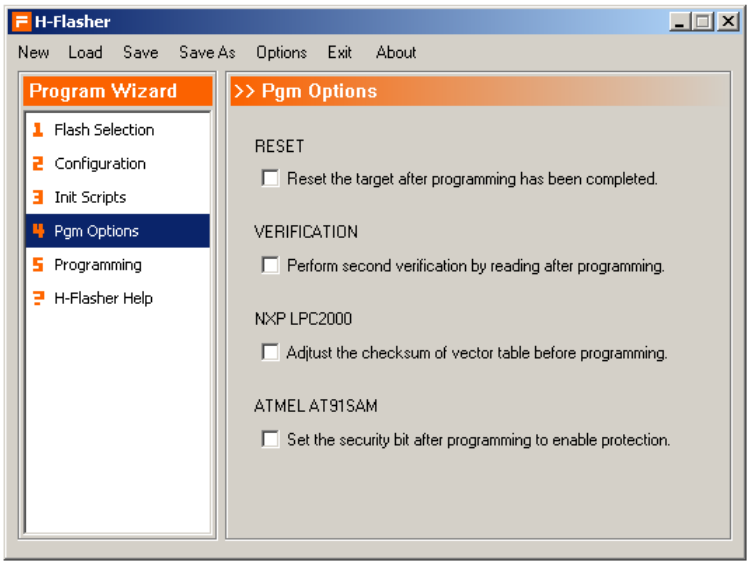

图 2-23 H-FLASHER 编程向导－Program Options

（5） Programming

Programming 是编程向导中的第五步，如下图所示。在这个步骤当中，用户可以对选定的目标 FLASH 芯片进行基本的操作：检测、烧写、验证、擦除和检查芯片是否为空等。

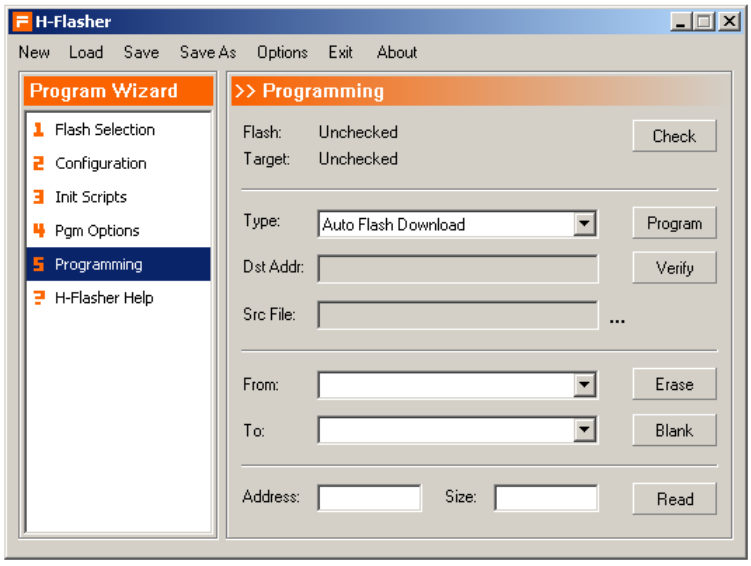

图 2-24 H-FLASHER 编程向导－Programming

（6） H-Flasher Help

在 H-FALSHER 的向导，还提供了一些基本的帮助信息，如果下图所示。这些基本的信息可以帮助用户了解每个步骤的设置。

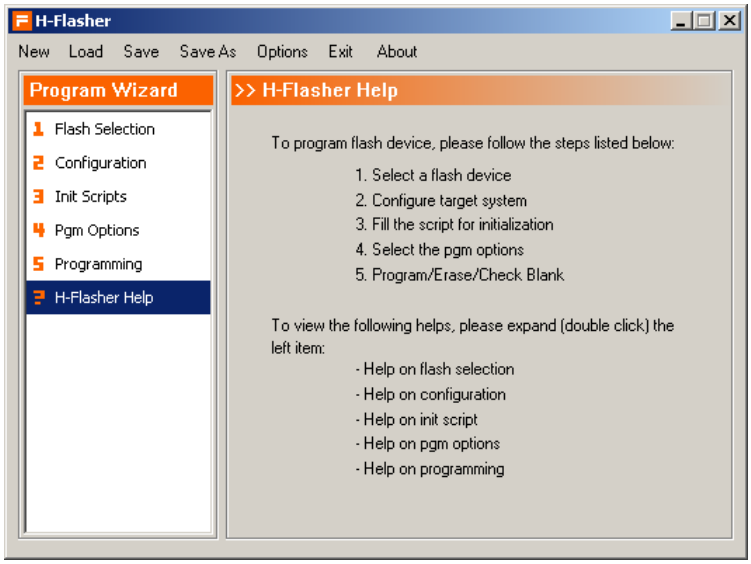

图 2-25 H-FLASHER 编程向导－Help

第三章 H-JTAG USB 仿真器

本章将简单介绍 H-JTAG USB 仿真器的硬件接口和驱动程序的安装.

3.1 USB 仿真器硬件接口

H-JTAG USB 仿真器是一款采用高速 USB2.0 接口的高性能 ARM 仿真器. 仿真器采用 USB 接口供电, 最高 TCK 时钟可以达到 15M Hz.

H-JTAG USB 仿真器的外观如图 3-1 所示. 左侧为 USB 接口, 右侧为标准的 20 针 JTAG 接口.

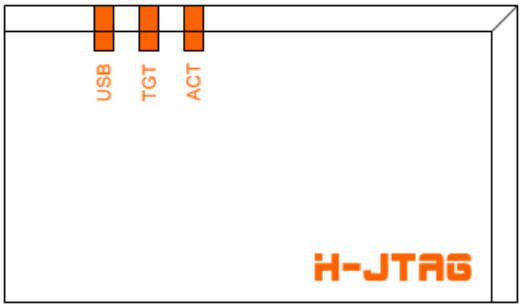

图 3-1 H-JTAG USB 仿真器外观

仿真器上有三个 LED 指示灯: USB, TGT 和 ACT. 这三个指示灯的作用如下:

A. USB – 标识 USB 电源, 用来识别 USB 是否连接;
B. TGT – 标识目标板的参考电压, 可以用来识别目标板连接是否正确;
C. ACT – 动作指示灯, 如果闪动表示正在对目标系统进行 JTAG 操作;

H-JTAG USB 仿真器的采用标准的 20 针 JTAG 接口, 接口信号的定义如下:

VREF	1	2	NC
nTRST	3	4	GND
TDI	5	6	GND
TMS	7	8	GND
TCK	9	10	GND
NC	11	12	GND
TDO	13	14	GND
nSRST	15	16	GND
NC	17	18	GND
NC	19	20	GND

图 3-2 H-JTG USB 仿真器 JITAG 接口信号定义

注意: H-JTAG USB 仿真器只支持高速 USB 2.0 接口.

3.2 安装 USB 仿真器驱动程序

在安装好 H-JTAG 程序后，在 H-JTAG 的安装目录下会有一个 Driver 目录，这个目录包含了 H-JTAG USB 仿真器的驱动程序. 下面介绍如何安装 H-JTAG USB 仿真器的驱动程序.

首先，将 H-JTAG USB 仿真器通过 USB 线连到计算机的 USB 接口上. 计算机随后会提示发现新硬件，并弹出如下图所示的硬件安装向导.

图 3-3 新硬件安装向导

在向导中，选择 Install from a list or specific location，如下图所示，然后点击 Next.

图 3-4 选择从列表安装

接下来，用户会看到如下图所示的搜索路径和选项对话框.在这个对话框中，选择 Search for the best driver in these locations. 取消 Search removable media，选择 Include this location in search. 同时点击 Browse 按钮，指向 H-JTAG 安装目录下的 Driver 文件夹. 然后点击 Next.

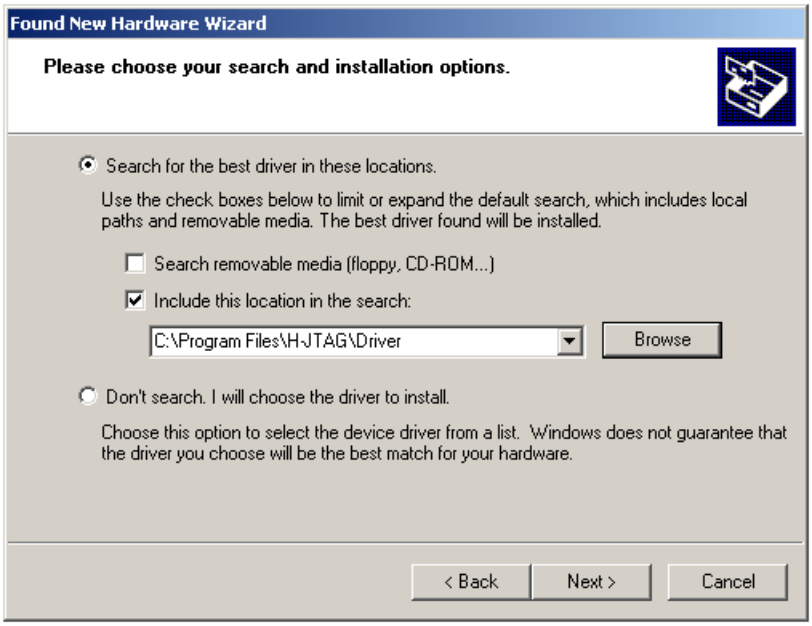

图 3-5 搜索和选项对话框

在经过短暂的搜索后，会弹出如下图所示的对话框,提示是否为新硬件安装 H-JTAG 驱动. 在下面的对话框中，点击 Continue Anyway，继续安装.

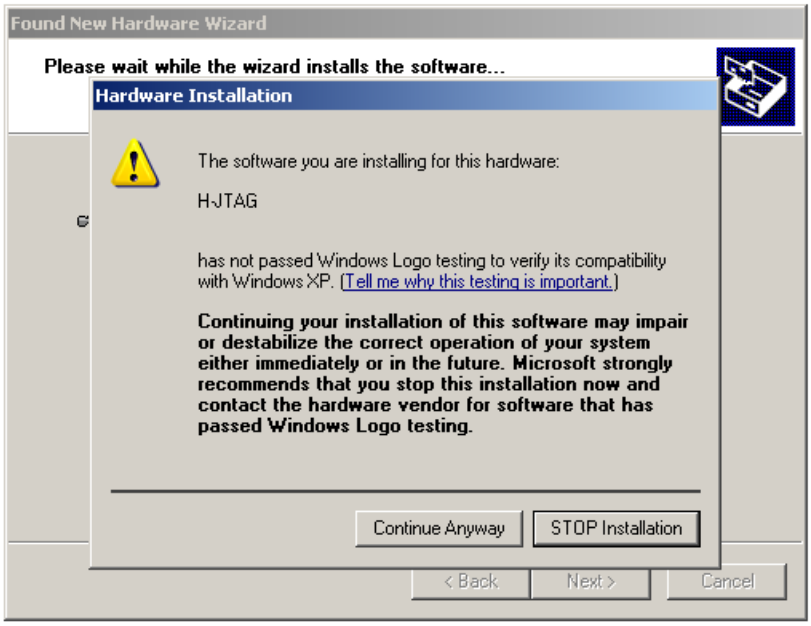

图 3-6 确认安装对话框

接下来，用户会看到如下图所示的对话框，提示用户 H-JTAG 驱动程序已经安装完成．点击 Finish，完成安装.

图 3-7 完成安装

在完成上面的安装后，用户可以打开硬件管理器．(我的电脑 -> 右键 -> 属性 -> 硬件 -> 设备管理)．如果硬件安装没有问题，在设备管理里，应该能看到 H-JTAG，如下图所示:

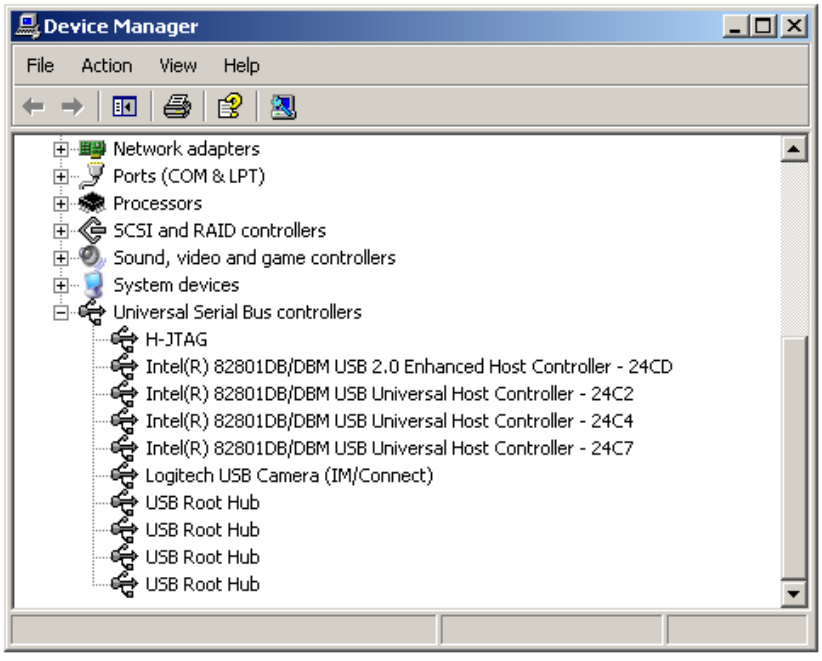

图 3-8 查看设备管理

如果在设备列表里看不到 H-JTAG，请重新安装 H-JTAG 的驱动程序.

第四章 H-JTAG 的使用和配置详解

这个章节将详细介绍 H-JTAG 的使用和配置。在介绍的过程当中，还结合实际提供了简单的例子，以帮助用户更好的理解。

4.1 检测调试目标

在检测前，请将 H-JTAG USB 仿真器连接到计算机 USB 接口或将 JTAG 调试小板连接到计算机并口，并和调试目标连上。在打开 H-JTAG 的时候，H-JTAG 会自动执行检测操作。用户也可以通过 H-JTAG 的菜单/工具栏进行检测。如果检测成功，H-JTAG 主界面将会显示检测到的芯片类型和 ID。如果检测失败，请检查 H-JTAG 设置和硬件连接。

4.2 复位调试目标

用户可用通过 H-JTAG 对调试目标进行复位。标准的 JTAG 接口，定义了两个独立的复位信号：系统复位（nSRST）和 JTAG 复位（nTRST）。通过这两个复位信号，可以分别执行系统复位和 JTAG 复位。用户可以选择执行系统复位、JTAG 复位，或是同时执行。H-JTAG 提供了相应的选项供用户选择，具体请参考 H-JTAG 常用选项。

提示：

因为有些并口 JTAG 调试板本身没有提供系统复位信号，所以不能通过 H-JTAG 执行系统复位，只能通过手动进行复位。有些并口 JTAG 调试板将系统复位和 JTAG 复位连接在一起，所以复位的时候相当于同时执行了系统复位和 JTAG 复位。

4.3 FLASH 自动下载

有些 ARM 芯片，片内集成了 FLASH 和 RAM，方便用户开发，省去了扩展外部存储的必要。针对这类芯片，例如 LPC2000 系列，AT91SAM7 系列，LUMINARY CORTEX-M3 系列和 STM32F 系列，H-JTAG 提供了 FLASH 自动下载功能。利用这个功能，调试的时候可以将程序直接烧写到 FLASH 中进行调试，就象直接装载到 RAM/SDRAM 里调试一样。要使用自动 FLASH 下载功能，请用户在 H-JTAG 里将 AUTO DOWNLOAD 选项打开（图 4-1），并在 H-FLASHER 里选择正确的目标芯片。设置好后，H-JTAG 会自动根据程序下载时的地址信息，区分那些部分需要下载到 RAM/SDRAM 中去，那些部分需要下载/烧写到 FLASH 中去。需要烧写到 FLASH 中去的程序部分，H-JTAG 会自动调用 H-FLASHER 完成烧写。

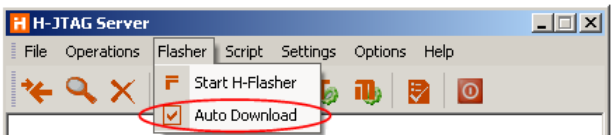

图 4-1 FLASH 自动下载设置

提示：

FLASH 自动下载一般只支持片内 FLASH，例如 LPC2000 和 AT91SAM7，因为这类芯片的存储系统相对来说比较简单和固定。对于支持较复杂存储配置的芯片，例如 MMU/REMAP 等，建议用户先用 H-FLASHER 将程序烧写到 FLASH 中里去，然后在进行调试。

4.4 设置初始化脚本

对于一般的系统来说，上电后都需要执行一些必要的初始化操作和配置，然后才能使用某些功能。一个最常见的初始化操作就是存储系统的配置。大部分情况下，FLASH 和片内 SRAM 在上电后都是可以直接访问，但片外 SDRAM 需要预先配置，才能正确访问。很多新手往往会碰到程序下载不正确的问题，最常见的是：为什么程序下载后看到的是乱码？为什么程序会跑飞？为什么跑不到 MAIN 函数？很多时候，都是因为没有对存储系统进行必要的初始化造成的。执行系统初始化有两种常用的方法。第一种方法是在 FLASH 芯片里烧一段初始化程序，上电后自动执行。这样，系统每次上电后目标系统就自动配置好了，可以直接进行调试。第二种方法就是通过初始化脚本来实现。用户根据数据手册编辑好初始化脚本，然后通过执行脚本以对系统进行初始化。有些调试器（DEBUGGER）提供了命令行窗口，用户可以通过命令行执行脚本命令。为了方便用户，H-JTAG 也提供了自动初始化的功能。要使用自动初始化功能，用户首先需要在 H-JTAG 里输入/装载初始化脚本，同时把 AUTO INIT 选项给打开（图 4-3）。这样，每次调试器（DEBUGGER）连接 H-JTAG 的时候，H-JTAG 都会自动执行用户指定的初始化脚本，对系统进行初始化。初始化脚本设置窗口如下图所示：

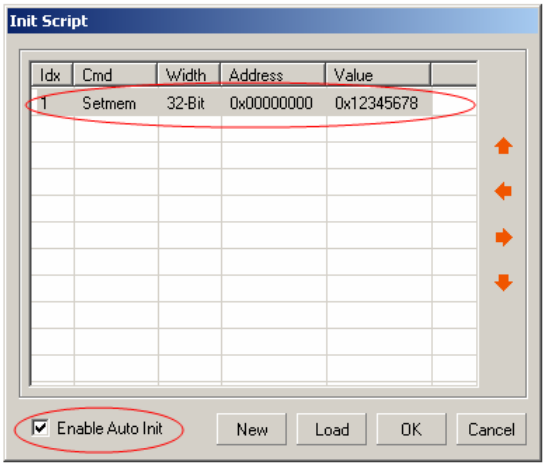

图 4-2 初始化脚本设置

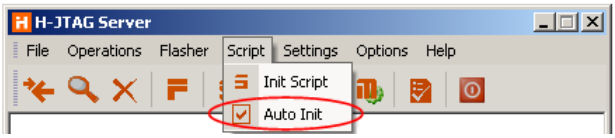

图 4-3 自动初始化功能

提示：

如果用户打开了 H-JTAG 的 AUTO INIT 选项，就必须指定初始化脚本，这样 H-JTAG 才知道如何执行初始化。如果打开了该选项，但没有指定初始化脚本，H-JTAG 会提示：“Can’t open specified init script”。

提示：

H-JTAG 脚本的具体定义，以及脚本的编辑，请参考第六章。

4.5 USB/LPT 接口选择

H-JTAG 支持并口调试板和 USB 接口 H-JTAG 仿真器。用户需要根据自己使用的硬件，选择 LPT 接口或是 USB 接口，如果图 4-4 所示。

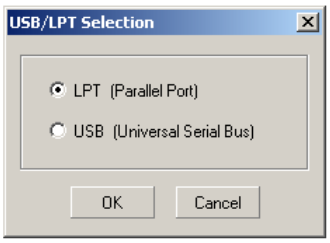

图 4-4 USB/LPT 接口选择

4.6 设置 JTAG

这个小节简单介绍了 JTAG 的接口定义，以及 JTAG 的配置。最后还提供了一个 JTAG 设置的例子，供用户参考。

4.6.1 JTAG 信号

JTAG 是由 IEEE 制定的一个测试标准，其标准号为 IEEE-1149，如果用户想详细了解该标准，请参考 IEEE 的 JTAG 标准。ARM JTAG 调试接口总共定义了 7 个基本的信号：TMS、TCK、 TDI、 TDO、 RTCK、 nSRST、 nTRST。对于 ARM7/9 的调试，TMS、 TCK、 TDI 和 TDO 为必选信号，而 RTCK、 nSRST、 nTRST 为可选信号。

注意：

XSCALE 的调试要求 JTAG 调试板提供 nSRST 和 nTRST 信号，并且必须是独立的。否则，无法进行调试。

4.6.2 JTAG 连接

H-JTAG 通过与并口/USB 连接的 JTAG 控制器来产生 JTAG 信号，并通过调试目标板的 JTAG 接口来控制 ARM 处理器。典型的连接如图 4-5 所示。其中 JTAG 接口是定义在 ARM 目标开发板上的，一般都采用标准的 20-PIN 或 14-PIN 的 JTAG 接口，不需要用户设置。用户可以使用 USB 接口的 H-JTAG 仿真器，或是基于并口的 JTAG 调试板。如果使用并口调试板，用户需要提供并口与 JTAG 调试板的具体连接。具体设置请参考后面的小节。

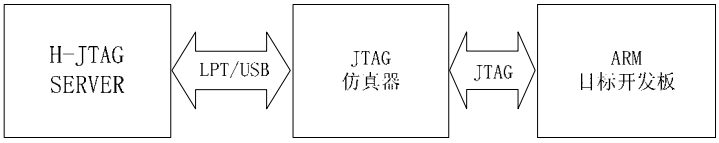

图 4-5 JTAG 连接

4.6.3 USB JTAG 设置

H-JTAG USB 仿真器支持不同的 TCK 速率（25K – 15M Hz），用户可以在 USB JTAG 设置对话框里选择合适的 TCK 速率。除了指定特定的 TCK 速率，用户也可以选择 AUTO TCK。当选择 AUTO TCK 的时候，H-JTAG 会通过测试选择适合目标系统的 TCK 速率。

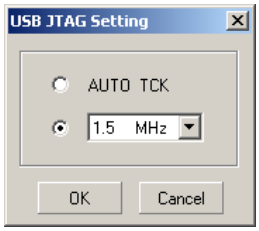

图 4-6 USB TCK 设置

提示：

TCK 速率的选择会直接影响调试速度。不同的目标系统支持的最高 TCK 速率不一样.。另外，TCK 频率和目标系统的时钟设置也有很大的关系。用户需要根据目标系统选择合适的 TCK 速率，否则调试会出错。选择 AUTO TCK 的时候，H-JTAG 有可能会设置过高的 TCK 速率，如果遇到此类情况，请手动选择低一些的 TCK 速率。

4.6.4 LPT JTAG 设置

简单的并口 JTAG 调试板并没有固定的电路图，就算是常用的 WIGGLER 和 SDT-JTAG，也没有固定的电路图。例如，有些 WIGGLER 支持系统复位信号（nSRST），而有些却没有。有些板子的系统复位信号和 JTAG 复位信号是独立的，有些却是连接在一起的。针对这样的情况，H-JTAG 提供了灵活的并口 JTAG 设置，以支持不同的并口 JTAG 调试板。用户需要根据自己使用的 JTAG 调试板进行设置，明确告诉 H-JTAG 并口与 JTAG 调试板是如何连接的。

并口提供了 8 位数据位 D0-D7，其方向为输出。我们可以利用这些数据位来输出 JTAG 控制信号：TMS、TCK、 TDI、 nSRST、 nTRST。并口还提供了几个状态位，其方向为输入。我们可以利用其中的一位来读取 TDO 的状态。所以，JTAG 的设置就是要告诉 H-JTAG，并口与 JTAG 调试板是如何连接的，并口数据位/状态位与 JTAG 信号是如何一一对应的。在提供了 nSRST/nTRST 信号的 JTAG 调试板上，复位信号有可能是经过反向三极管来产生反向电平的。所以，用户也需要在配置的时候告诉 H-JTAG 复位信号是否是反向的。这样 H-JTAG 才能正确的执行复位操作。

下面我们来看一个例子。假设并口 JTAG 调试板的电路图如图 4-7 所示。从图中我们可以发现并口和 JTAG 接口的连接如下，其中 nTRST 信号经过了一个反向三极管，而 nSRST 信号根本没有提供。

TMS	➡	并口 D1 (PIN3)
TCK	➡	并口 D2 (PIN4)
TDI	➡	并口 D3 (PIN5)
TDO	➡	并口 BUSY (PIN11)
nTRST	⬅	并口 D0 (PIN2) 反向
nSRST	X	没有提供

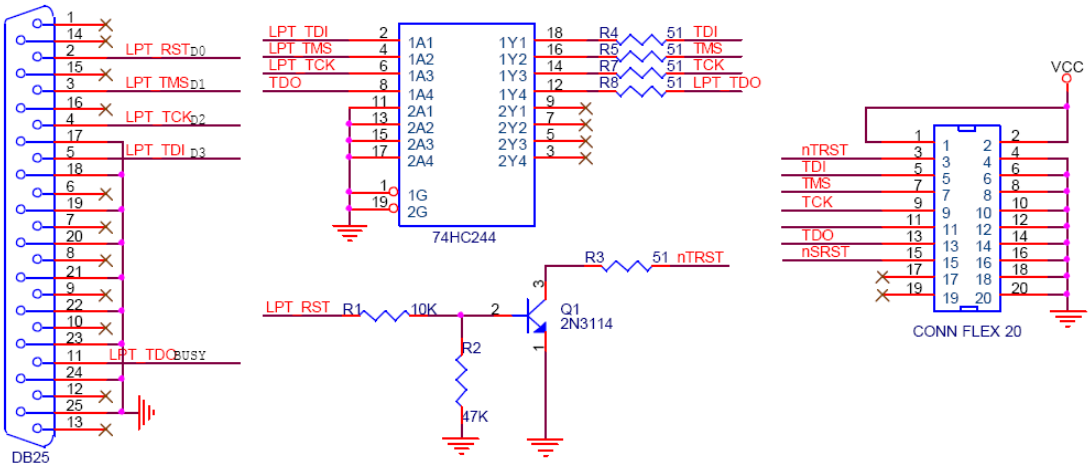

图 4-7 并口 JTAG 调试板电路图

根据图 4-7 所示的 JTAG 调试板的电路图和上面的分析，JTAG 设置可以采用下面的任何一种设置。下面的设置明确的告诉了 H-JTAG 并口和 JTAG 调试板是如何连接的。

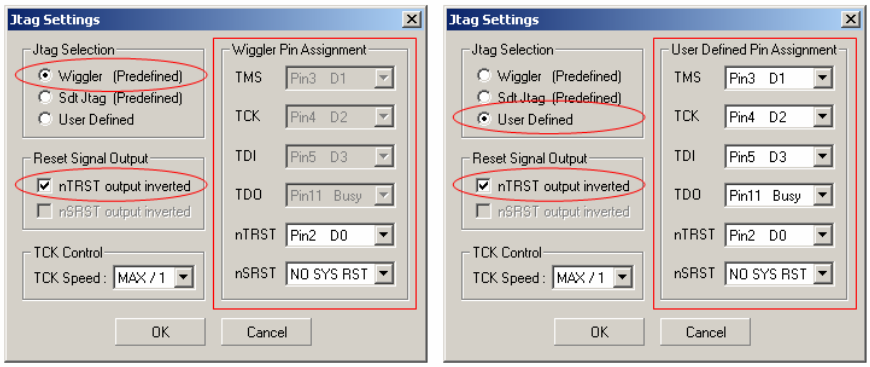

图 4-8 并口 JTAG 设置实例

上面我们给出了一个并口 JTAG 配置的实例，供用户参考。在实际应用当中，用户需要根据自己使用的并口 JTAG 调试板的电路图，进行相应的设置。

在并口 JTAG 设置中，用户也可以根据需要选择不同的 TCK 速度，设置窗口如下图所示。因为并口的速度本身比较慢，大部分情况下，用户不需要降低 TCK 的速度，所以建议用户选择默认设置 MAX/1。

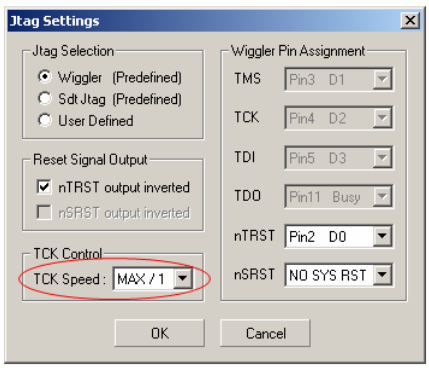

图 4-9 设置并口 TCK 速度

4.7 设置并口

对于大部分计算机，并口的默认地址都是 0x378，但也有些计算机的并口地址不是 0x378。H-JTAG 提供了并口设置窗口，用户可以根据自己的情况设定并口地址。并口设置窗口如下图所示。在并口设置窗口上还有一个端口测试按钮，用户可以用它对并口做简单的读写测试。

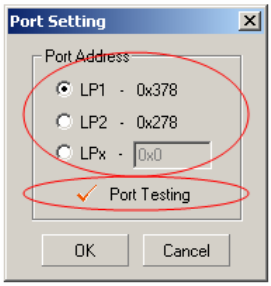

图 4-10 并口设置

4.8 设置调试目标

H-JTAG 通过 JTAG 读取芯片的 ID，并根据此 ID 来判断调试目标采用的是什么 ARM 内核。H-JTAG 可以识别常用的大部分芯片。如果 H-JTAG 不能识别调试目标，用户可以在 TARGET SETTINGS 里手动指定 ARM 内核。TARGET SETTINGS 的设置窗口如图 4-11 所示。

一般的 ARM 芯片，都同时支持大小端（Little Endian/Big Endian）。不同的方式，数据和指令的存储是完全不同的，具体的请参考数据手册。如果大小端设置不正确，调试的时候肯定会出问题。用户需要在 TARGET SETTINGS 里面指定调试目标的大小端。大小端的选择如下图所示：

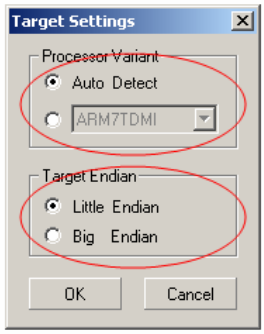

图 4-11 调试目标设置

4.9 添加芯片 ID

上面我们提到，H-JTAG 根据芯片 ID 来判断芯片使用的是什么 ARM 内核。对于不能识别的芯片，用户可以手动指定其类型。同时，用户也可以将新的 ID 添加到 H-JTAG 的芯片列表当中去。这样 H-JTAG 就可以自动检测该芯片。H-JTAG 提供了 TARGET MANAGER，用户可以利用这个管理器添加新的芯片 ID，或删除已经存在的芯片 ID。要添加新的芯片 ID，用户需要在管理器中输入新 ID，并指定该 ID 对应的 ARM 内核。完成后，H-JTAG 会自动更新芯片列表。H-JTAG 的 TARGET MANAGER 如下图所示：

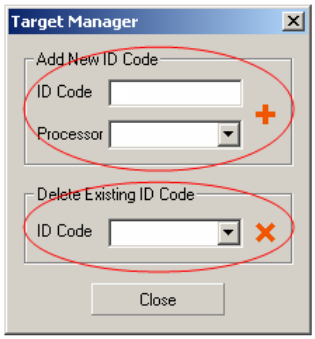

图 4-12 调试目标管理器

提示：

根据 IEEE 1149 标准，芯片 ID 为 32 位，并且 ID 的最低位必须为 1。用户可以通过这点来判断芯片 ID 的有效性。同时，如果用户有 H-JTAG 不能识别的芯片，欢迎你把芯片 ID 和型号发给我们，我们会及时更新 H-JTAG 的芯片列表。

4.10 TAP 设置

对大部分的 ARM 芯片而言，ARM JTAG 扫描链是独立的。对于这样的芯片，TAP CONFIGURATION 应该采用默认设置，如下图所示。下图表示在 ARM 扫描链的前面和后面都没有串接别的扫描链。

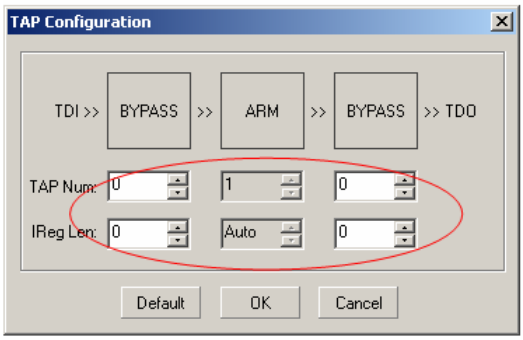

图 4-13 TAP 默认设置

有些 ARM 芯片的 JTAG 扫描链是和芯片内部集成的其它部件的 JTAG 扫描链串接在一起的。这样的芯片比较特殊。对于这类特殊的芯片，用户需要在 TAP CONFIGURATION 里进行相应的设置。例如，ST 公司的 STR91xF 芯片，其内部的 JTAG 扫描链如图 4-14 所示。从图中可以看到，STR91xF 芯片内部的扫描链包括三部分：TAP#1，TAP#2 和 TAP#3。其中 TAP#2 是 ARM 调试需要访问的扫描链。在 ARM 扫描链的前面和后面都串接了别的扫描链，他们的指令寄存器的长度分别为 5-Bit 和 8-Bit。对 STR91xF 芯片而言，TAP CONFIGURATION 应该设置成图 4-15 所示。图 4-15 的设置告诉 H-JTAG，在 ARM 扫描链之前有一条别的扫描链，指令寄存器的长度为 5-Bit，在 ARM 扫描链的后面也有一条别的扫描链，其指令寄存器的长度为 8-Bit。这样，H-JTAG 就知道该如何正确访问 STR91xF 内部的 ARM 调试扫描链。

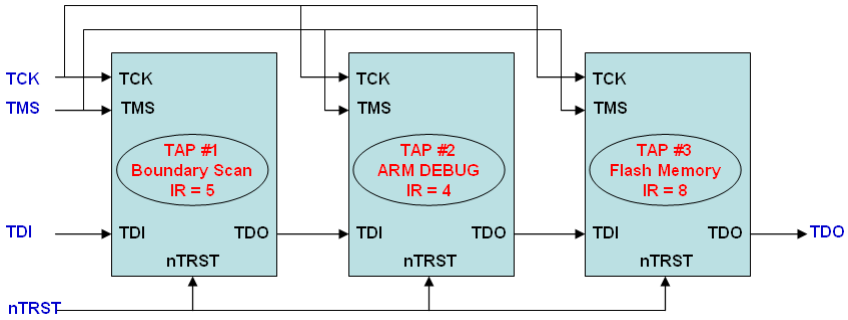

图 4-14 STR91xF 扫描链内部结构

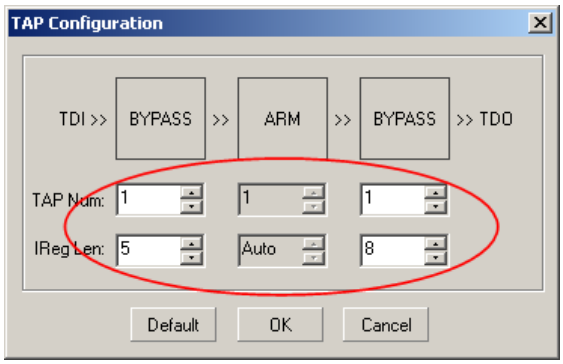

图 4-15 TAP 设置（STR91xF）

4.11 H-JTAG 常用选项

H-JTAG 提供了一些常用的选项供用户选择。用户可以通过选项设置主窗口进行设置，也可以通过 OPTIONS 菜单进行快速设置。选项设置主窗口如下图所示。用户可以在左侧窗口选择需要设置的选项，然后在右边窗口进行设置。

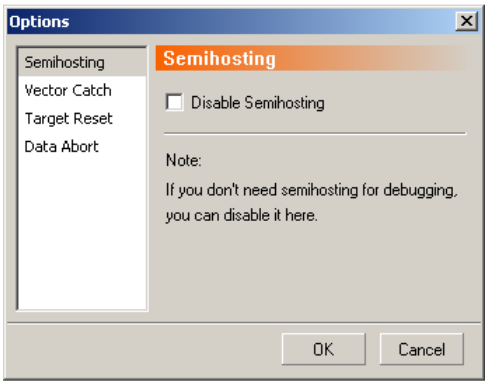

图 4-16 选项设置主窗口

H-JTAG 提供了五个常用选项：

■ VECTOR CATCH 设置

H-JTAG 集中管理 VECTOR CATCH，如图 4-17 所示。在 VECTOR CATCH 设置里，用户可以选择在调试过程中需要捕获的中断或异常。如果全局 DISABLE VECTOR CATCH 选项被使能，VECTOR CATCH 的设置将被忽略。

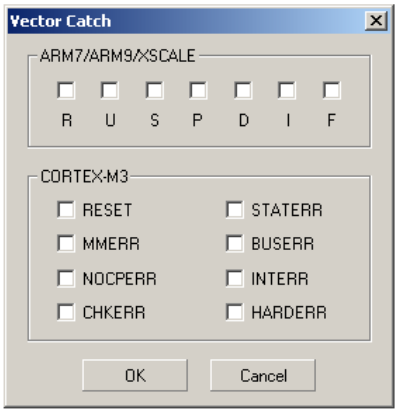

图 4-17 VECTOR CATCH 设置

■ SEMIHOSTING

Semihosting 是一种辅助调试机制，用来实现主机与目标开发板的通信。但需要仿真器和目标程序的配合和支持，而且 Semihosting 只能用在调试上，在实际的产品当中是不能使用的。因为 Semihosting 会占用断点资源，建议用户在调试的时候禁用掉该功能。

■ VECTOR CATCH

Vector Catch 是用来捕获异常的。如果打开了该选项，并在调试软件里做了相应的设置，ARM 处理器出现异常的时候，调试软件就会提示用户出现了异常。Vector Catch 也会占用断点资源，建议用户在调试的时候，禁用掉该功能。

■ TARGET RESET

在该选项里，用户可以选择如何对调试目标进行复位。用户可以选择只用 nTRST 信号，或是同时使用两个信号对系统执行复位。用户还可以选择每当调试软件连接到 H-JTAG SERVER 的时候，自动执行一个复位操作。另外, 用户也可以指定复位等待时间. 这样 H-JTAG 会在执行完复位操作后, 等待用户指定的时间, 再停止 MCU 的运行.

提示：

复位操作依赖于用户具体使用的 JTAG 调试板。如果 JTAG 调试板没有系统复位信号，H-JTAG 将无法执行系统复位。

■ DATA ABORT

在调试的过程当中，即使在 CPU 停止运行的时候，也需要对存储系统进行读写，例如设置/清除断点。在对存储进行读写的时候，有可能会出现数据异常。例如，访问未定义的存储空间，或是访问受保护的存储空间。如果用户打开了 REPORT DATA ABORT 选项，H-JTAG 在检测到数据异常的时候，将会提示用户。否则，H-JTAG 只是在内部进行处理，而不会提示用户。用户可以根据自己的需要进行选择。

4.12 检查软件更新

通过 H-JTAG 的 CHECK UPDATE 菜单，用户可以检查是否有新的版本可以下载。如果有更新的版本，H-JTAG 会提示用户下载。用户也可以访问 H-JTAG 的主页以获得更多的更新信息。

第五章 H-FLASHER 的使用和配置详解

这个章节详细介绍了 H-FLASHER 的使用和配置。同时还提供了两个配置实例，供用户参考。

5.1 H-FLASHER 工作流程

H-FLASHER 的工作原理很简单，其流程如图 5-1 所示。H-FLASHER 的工作流程分为四个主要的步骤：执行初始化脚本、下载 FLASH 烧写驱动、检查 FLASH ID 和对 FLASH 执行用户指定的操作。如果任何一个步骤出错，H-FLASHER 都会提示错误，中断当前的操作。

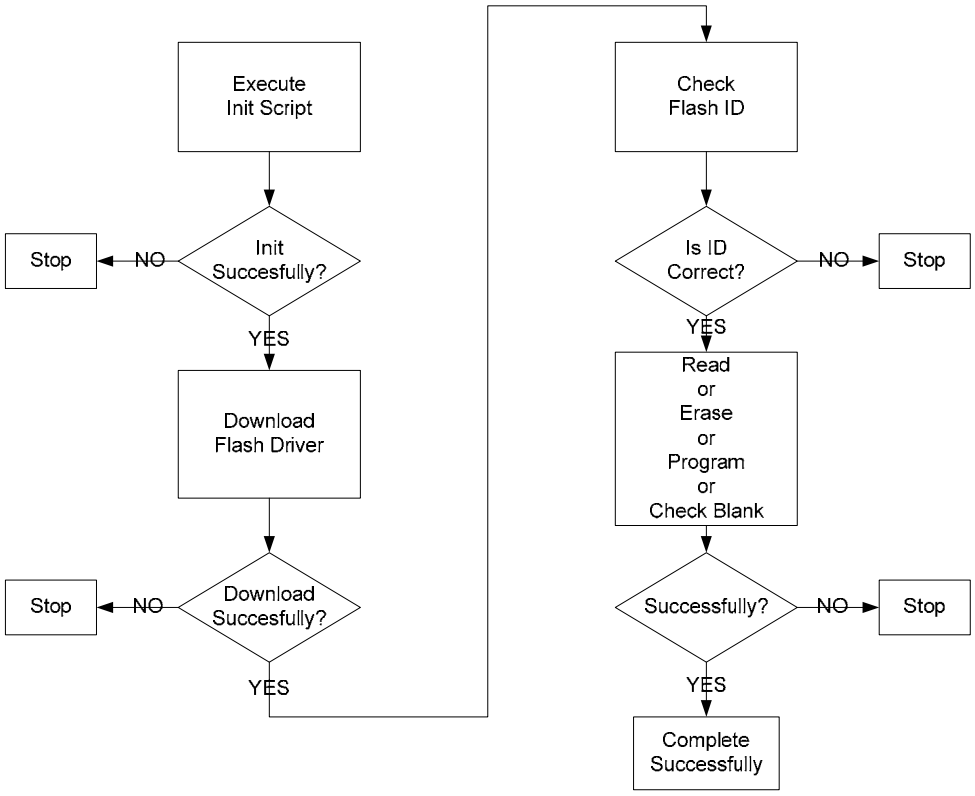

图 5-1 H-FLASHRE 工作流程

5.1.1 Execute Init Script

如果用户指定了初始化脚本，H-FLASHER 首先会执行用户指定的初始化脚本。如果用户没有指定初始化脚本，或是不需要用户提供初始化脚本，H-FLASHER 会直接跳过这一步。在初始化的时候，如果脚本有错或执行脚本的时候出错，H-FLASHER 会停止当前的操作，并提示错误。如果初始化成功，则准备下载 FLASH 驱动。

5.1.2 Download Flash Driver

初始化完成后，H-FLASHER 会根据用户选择的 FLASH 型号和位宽，查找相应的驱动。然后把驱动下载到用户指定的 SRAM/SDRAM 中去。如果下载成功，则进入下一步；否则停止当前的操作，并提示用户：Can’t download driver to specified address。

5.1.3 Check Flash ID

FLASH 驱动程序下载成功后，H-FLASHER 会马上读取 FLASH 的 ID。这样做有两个作用，一是看看在指定的地址上是否能访问 FLASH；二是通过 ID 来判断用户选择的 FLASH 芯片型号和目标系统上的 FLASH 芯片是否一致。

5.1.4 Read/ Erase/ Program/ Verify/Check Blank

在前面的步骤都成功执行后，H-FLASHER 就会根据用户的请求对 FLASH 芯片执行相应的操作，例如：读取，擦除，编程, 校验和检查 FLASH 是否为空。操作完成后，提示用户操作是否成功。

5.2 H-FLASHER 编程向导

H-FLASHER 提供了编程向导，以方便用户配置和烧写 FLASH。用户可以一步一步的按照编程向导提供的五个步骤进行设置。在介绍了 H-FLASHER 的基本工作流程后，我们一起来看看在编程向导的每个步骤中，该如何进行设置。

5.2.1 Flash Section

在这个步骤当中，用户需要在 FLASH 列表里选择相应的 FLASH 芯片型号，并确定 H-FLASHER 提供的芯片信息和目标系统上的 FLASH 芯片的信息是一致的。因为不同的 FLASH 芯片定义了不同的操作命令，所以正确选择 FLASH 芯片型号很重要。

5.2.2 Configuration

在这个步骤当中，用户需要提供基本的配置信息，包括 FLASH 的位宽芯片数量，FLASH 的起始地址，RAM 的起始地址, 外部晶振的频率和不同阶段使用的 TCK 频率.

(1) 位宽和 FLASH 芯片数量

有些片外 FLASH 的读写操作同时支持不同的位宽，例如 8-BIT、 16-BIT 或 32-BIT。如果 FLASH 芯片支持不同的读写位宽，需要用户根据实际的硬件连接选择 FLASH 采用的位宽。因为不同的连接，FLASH 工作的位宽不一样，FLASH 烧写驱动也有所区别。如果目标 FLASH 芯片只支持一种读写方式，则不需要指定。另外, 在设计当中, 有时候也会使用 2 片或多片 FLASH 组成一个容量更大的存储模块. 对于多芯片的情况, 用户需要正确指定芯片的数量.

(2) FLASH 起始地址

H-FLASHER 需要知道 FLASH 的起始地址，才能对 FLASH 进行正确的操作。所以，用户需要指定 FLASH 的起始地址。H-FLASHER 会根据用户指定的起始地址和 FLASH 的容量自动计算 FLASH 的有效地址空间：FLASH START ADDRESS ～ (FLASH START ADDRESS + SIZE – 1)。对 FLASH 进行写操作的时候，如果烧写目的地址不在这个范围内，H-FLASHER 将会提示 OUT OF RANGE 的错误。大部分情况下，FLASH 都可以从地址 0X0 访问，而不需要特别的初始化。但有些目标系统支持 REMAP，可以把 FLASH 配置到不同的地址。对于这样的系统，用户需要保证设置的 FLASH 起始地址与提供的初始化脚本是相一致的，以免出错。总之，如果有需要，请提供初始化脚本，对存储系统进行初始化，保证 H-FLASHER 可以在用户指定的地址访问目标 FLASH。

(3) RAM 起始地址

在烧写的时候，FLASH 驱动需要使用 4KByte 的 RAM 空间，所以需要用户指定一段>=4KByte 的 RAM 空间。在用户指定 RAM 开始地址后，H-FLASHER 会把 FLASH 驱动下载到 RAM START ADDRESS ~ (RAM START ADDRESS + 4K – 1)的范围内。FLASH 驱动可以下载到片内 SRAM 或是片外 SDRAM。如果目标系统有片内 SRAM， 建议用户使用片内 SRAM，因为片内 SRAM 的速度比外部 SDRAM 快。在设置初始化脚本的时候，请提供必需的初始化脚本，对存储系统进行初始化，保证用户指定的 RAM 空间是可以被正确访问的。

(4) XTAL

对有些芯片而言，H-FLASHER 需要知道目标系统外接晶振的频率。FLASH 驱动会根据用户指定的 XTAL 频率对系统时钟进行配置, 以获得更好的烧写速度. 在选定 FLASH 芯片型号后，如果 XTAL 对应的编辑框处于禁用状态，则说明不需要用户指定 XTAL。

(5) INIT TCK 和 PGM TCK

INIT TCK 和 PGM TCK 分别指定了初始化期间使用的 TCK 频率和在烧写过程中使用的 TCK 频率. 在对目标系统执行初始化之前, 目标系统支持的 TCK 频率可能比较低. 在对目标系统执行初始化后,(例如配置了系统时钟), 目标系统可以支持更高的 TCK 频率. 这样用户就可以指在不同阶段使用的 TCK 频率. 让 H-FLASHER 在初始化阶段使用较低的 INIT TCK, 以顺利完成初始化操作. 在对 FLASH 进行操作的过程中, 使用较高的 PGM TCK, 以达到更好的速度.

注意:

INIT TCK 和 PGM TCK 设置只对 H-JTAG USB 仿真器有效. 并且只有在同时指定了 2 个 TCK 频率的时候生效.

5.2.3 Init Script

在这个步骤当中，用户需要提供初始化脚本，用以对目标系统进行初始化。H-FLASHER 专门提供了脚本编辑窗口，用户可以添加和删除脚本，并对脚本进行排序。关于初始化脚本的具体定义和编辑，请参考第六章。

对片内 FLASH 而言，一般不需要用户提供初始化脚本。而对片外 FLASH，一般都需要用户提供初始化脚本。初始化脚本主要有两个作用：一是设置系统时钟，这可以保证 ARM 处理器的运行速度；二是配置调试目标的存储系统，以保证 H-FLASHER 可以正确的对 FLASH 进行操作，同时也保证 FLASH 驱动可以正确下载到目标系统的 SRAM/SDRAM 中去。

初始化脚本直接影响到 FLASH 驱动是否能成功下载。如果 H-FLASHER 提示错误：Can’t download driver to specified address，大部分情况下都是用户没有提供初始化脚本，或是初始化脚本有问题。要提供正确的初始化脚本，要求用户对目标系统有一定的了解。建议用户仔细阅读目标芯片的数据手册，特别是存储设置部分。

提示:

在使用 H-JTAG 仿真器的时候, 为了提高性能, 建议用户添加系统时钟配置脚本. 这样可以有效的提高 TCK 时钟和 FLASH 烧写速度.

5.2.4 Program Options

H-Flasher 提供了些常用的选项，例如复位操作，二次验证和加密等。在这个步骤中，用户可以根据自己的需要进行选择。

注意：

这些选项只在非自动下载模式下起作用。在自动下载模式下，H-Flasher 将忽略这些选项。

5.2.5 Programming

在这个步骤当中，用户可以对 FLASH 芯片执行以下的操作：检查 FLASH 及其目标系统的基本信息、对 FLASH 进行烧写, 擦除, 校验和检查 FLASH 是否为空。

(1) Check

这个操作可以读取目标系统上的 FLASH 的 ID 及其处理器的基本信息。用户可以使用 Check 功能来检查用户的配置是否正确，读取的目标系统的信息是否正确。

(2) Program

H-FLASHER 提供了三种烧写方式供用户选择：Auto Flash Download，Intel HEX Format，和 Plain Binary Format。在烧写的时候，烧写的数据/程序和目的地址都是必须的。这三种方式的唯一区别在于这些信息由哪里获得。

A - Auto Flash Download

采用这种方式的时候，用户不需要指定源文件和目的地址。要烧写的数据和烧写的目的地址都是由 H-JTAG 提供给 H-FLASHER 的。

B - Intel HEX Format

因为 HEX 文件本身包含了地址信息。所以，选择这种文件格式的时候，用户只需要指定 HEX 文件的路径，而不需要指定烧写的目的地址。H-FLASHER 会自动从 HEX 文件中提取地址信息。

C - Plain Binary Format

Binary 文件本身不包含除了程序/数据外的任何其它信息。所以，选择这种文件格式的时候，用户需要同时指定二进制文件的路径和烧写的目的地址。

(3) Verify

这个操作可以用来验证烧写是否正确. Verify 操作会将 FLASH 的内容读取回来, 然后和用户指定的文件进行比较. 以确定烧写是否成功.

(4) Erase & Check Blank

用户可以对 FLASH 进行擦除和检查 FLASH 是否为空。在执行这两个操作的时候，用户可以选择对整块 FLASH 执行操作，或是通过 FROM 与 TO 下拉框选择操作的地址范围。

(5) Read

H-Flasher 还提供了读取操作，用以读取存储系统的内容。在执行读取操作的时候，用户需要同时指定开始地址和长度。长度的单位是 BYTE。

5.3 常见提示

- **提示-1：**
  在设置的时候，如果有那个输入框或是下拉框是灰色的或是不可编辑的，说明该选项只有一个选择，不需要用户设定。

- **提示-2：**
  在用户配置好各个选项后，用户可以将配置保存为 HFC 文件。在需要的时候，可以在 H-FLASHER 里直接装载 HFC 文件，省去了每次都需要配置的麻烦。

- **提示-3：**
  烧写前，H-FLASHER 会自动执行必要的擦除操作。所以用户不需要在烧写的时候对 FLASH 芯片先执行擦除操作。

- **提示-4：**
  FLASH 擦除操作都是以块（Sector）为最小单位的，一个块的大小通常都是大于一个 BYTE。为了避免数据丢失，H-FLASHER 提供了自动备份和恢复机制。在执行擦除前，H-FLASHER 会备份部分数据，并在烧写的时候自动恢复这部分数据。通过这个机制，可以避免烧写区域外的数据被改写。

- **提示-5：**
  烧写的时候，如果用户看到错误提示：Destination flash address is out of range，说明用户指定的目的地址不在 FLASH 的有效地址范围内，或是 HEX 文件中提取的地址不在 FLASH 的有效地址范围内。请用户检查地址设置是否正确，或是 HEX 文件的地址是否正确。

- **提示-6：**
  有些 FLASH 芯片，随着新版本的推出，其芯片 ID 有可能会更改。如果发现这种情况，请用户和我们联系，我们会提供新的 FLASH 驱动程序。

- **提示-7：**
  如果对 FLASH 进行操作的时候出错，请检查配置是否正确。如果配置正确，但还是出现错误，请和我们联系，我们将会分析问题出在那里。必要的时候，我们会提供新的 FLASH 烧写驱动。

5.4 烧写实例 1 – AT91SAM7X256

AT91SAM7X256 是 ATMEL 公司的一款基于 ARM7 的芯片。该芯片带有 256KByte 的片内 FLASH。下面我们将介绍利用 H-FLASHER 的编程向导如何来配置和烧写 AT91SAM7X256。

5.4.1 Flash Selection

在向导的第一步当中，选择芯片型号。在这个例子当中，选择芯片 AT91SAM7X256，如下图所示：

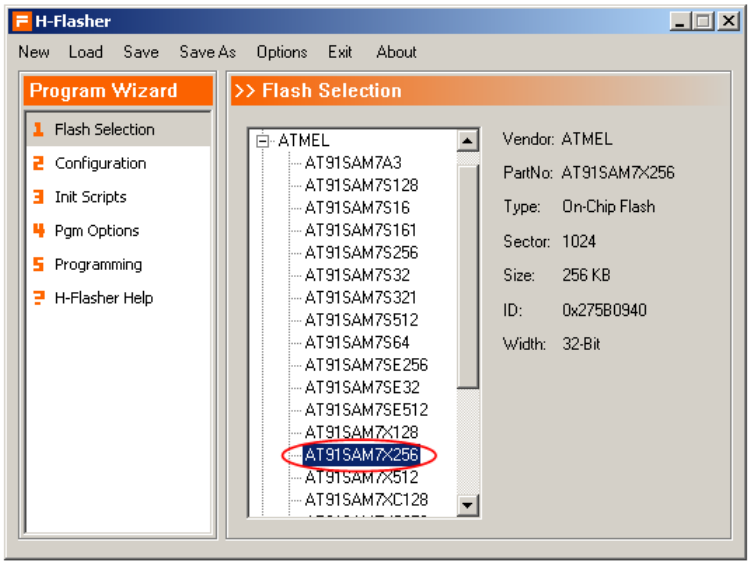

图 5-2 选择 AT91SAM7X256

5.4.2 Configuration

在向导的第二步中，对存储进行配置。因为烧写的是 AT91SAM7X256 的片内 FLASH，而且这块芯片的位宽、FLASH 起始地址和 RAM 起始地址都是固定的，所以用户不需要进行设置，采用默认的值就可以了。用户也不需要指定外部晶振的频率。设置如下图所示：

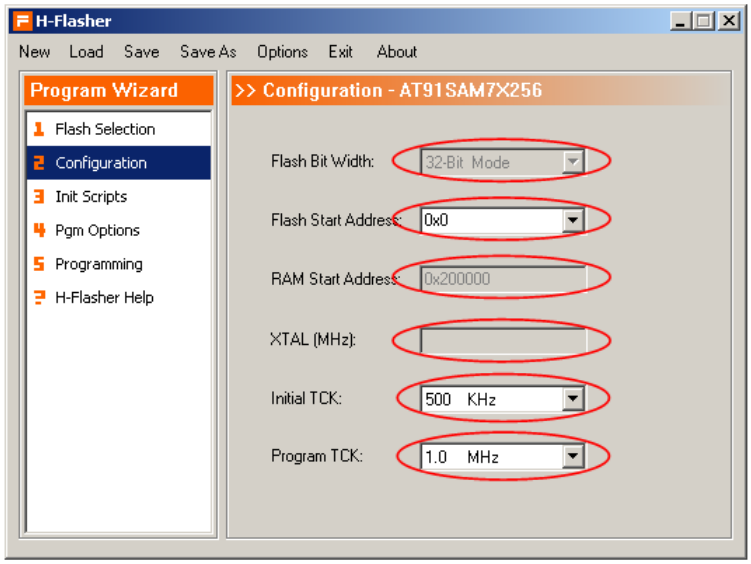

图 5-3 配置设置

5.4.3 Init Script

在向导的第三步中，设置初始化脚本。因为烧写的是 AT91SAM7X256 的片内 FLASH，而且 AT91SAM7X256 的烧写驱动当中已经包括了初始化部分，所以不需要用户提供初始化脚本。在这种情况下，脚本编辑的按钮全都是禁用的。如下图所示：

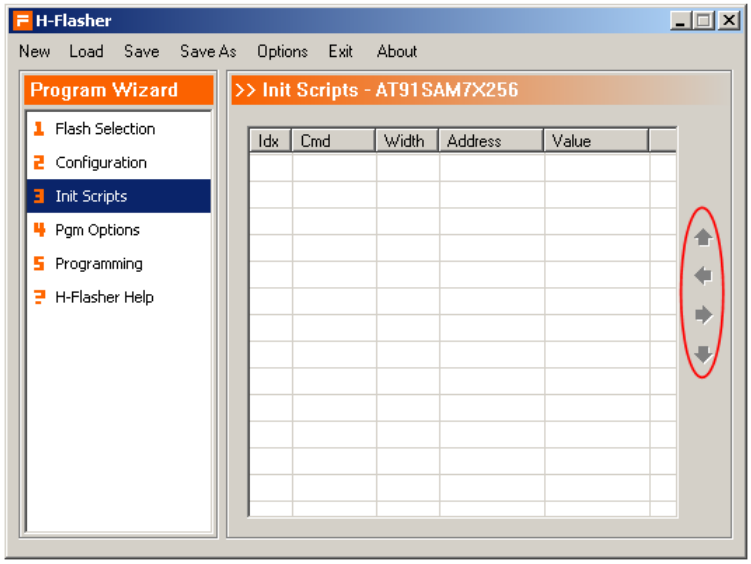

图 5-4 设置初始化脚本

5.4.4 Program Options

在向导的第四步中，用户可以选择不同的选项。在这里，我们选择了在烧写后执行系统复位，并执行二次验证，如下图所示：

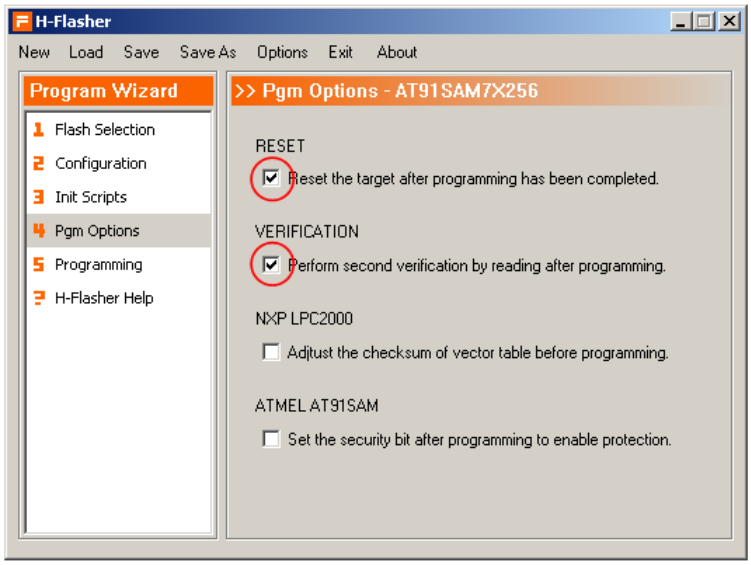

图 5-5 编程选项

5.4.5 Programming

在配置好后，在编程向导的第五步中，就可以对 FLASH 执行不同的操作了。首先，可以试一试 CHECK 操作。通过 CHECK 操作，用户可以大概判断前面的设置是否正确。在本例中，CHECK 的结果如图 5-6 所示。由图中显示的信息可以看出，前面的配置是正确的。

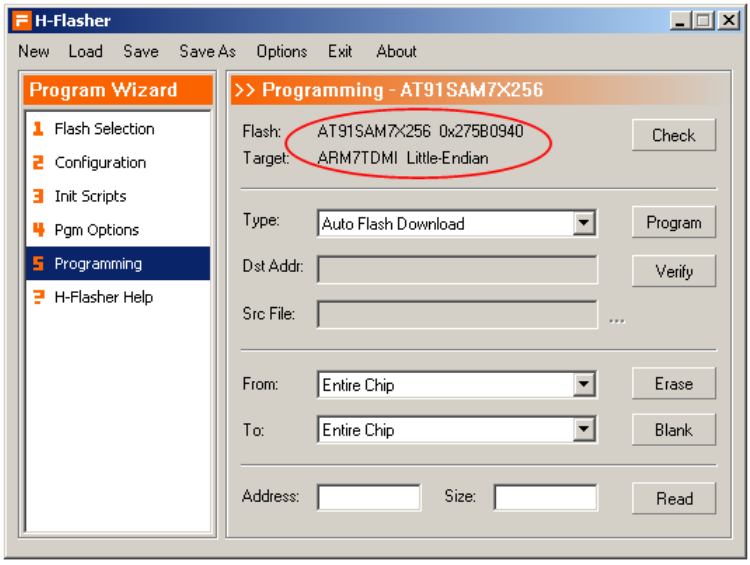

图 5-6 CHECK 操作结果

下面尝试烧写一个二进制文件。在本例中，选择文件格式为二进制，源文件为 C:\TEST.bin，烧写的目的地址为 0x0，设置如图 5-7 所示。设置好后，点击 PROGRAM 按钮开始烧写。烧写过程当中，用户会看到烧写文件的大小，平均烧写速度，当前的进度等信息。烧写完成后，H-FLASHER 会提示烧写并验证成功，如果图 5-8 所示。

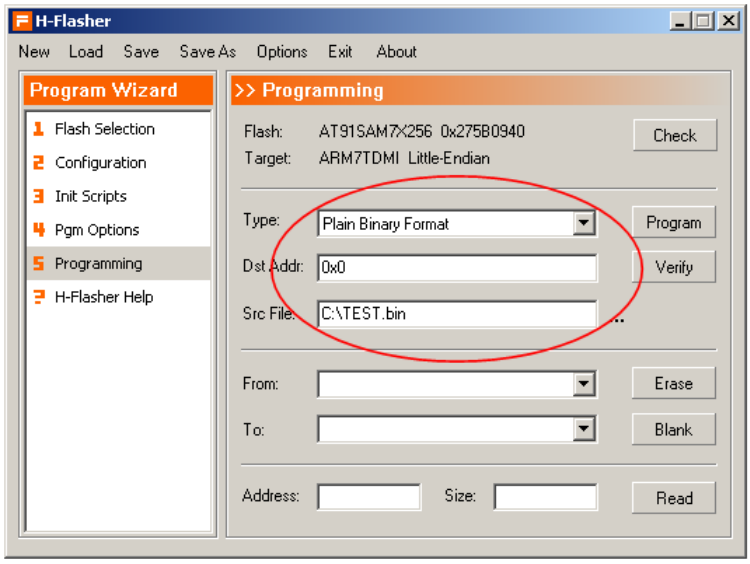

图 5-7 烧写设置

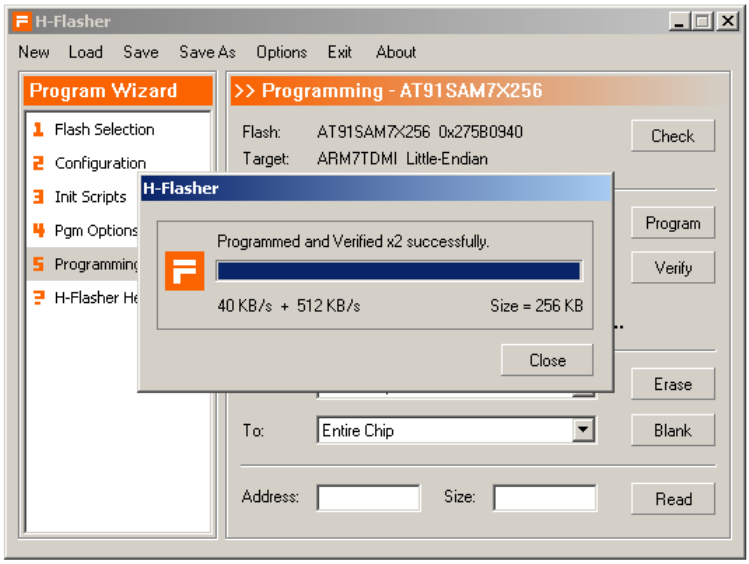

图 5-8 烧写完成

5.4.6 保存设置

用户可以将上面的设置保存为 HFC 配置文件，供以后使用。在以后的使用中，可以直接在 H-FLASHER 里装载 HFC 文件，省去了每次都必须进行设置的麻烦。

5.5 烧写实例 2 - LPC2210 + SST39VF1601

LPC2210 是 NXP 公司（前 PHILIPS 半导体）的一款 ARM7 芯片。该芯片有 16KB 的片内 SRAM，但没有片内 FLASH。该芯片有 4 个外部 MEMORY BANK，可用用来扩展外部 FLASH 和外部 SDRAM。在本例中，假设 BANK0 用来扩展外部 FLASH，型号为 SST39VF1601，BANK0 的地址范围为：0x80000000~0x80FFFFFF；BANK1 用来扩展外部 SDRAM，BANK1 的地址范围为：0x81000000~0x81FFFFFF。下面我们将介绍如何对 SST39VF1601 进行烧写。

5.5.1 Flash Selection

在向导的第一步当中，选择芯片型号。在这个例子当中，我们需要选择芯片 SST39VF1601，如下图所示：

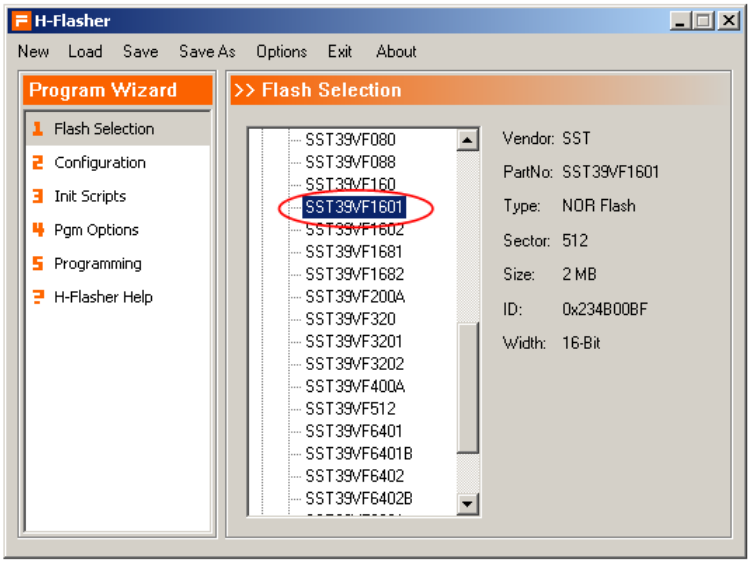

图 5-9 选择 SST39VF1601

5.5.2 Configuration

在向导的第二步中，对存储进行配置。SST39VF1601 只支持 16-BIT 模式，所以位宽采用默认设置。FLASH 是利用 BANK0 扩展的，而 BANK0 的起始地址是 0x80000000，所以 FLASH 的起始地址应该是 0x80000000。在指定 RAM 空间的时候，我们可用使用片内 SRAM 或是片外 SDRAM，在本例中，我们选择使用外部 SDRAM。外部 SDRAM 是利用 BANK1 扩展的，而 BANK1 的起始地址是 0x81000000，所以 RAM 的起始地址可以设为 0x81000000。在这个例子里，用户也不需要指定外部晶振的频率。最终的设置如图 5-10 所示：

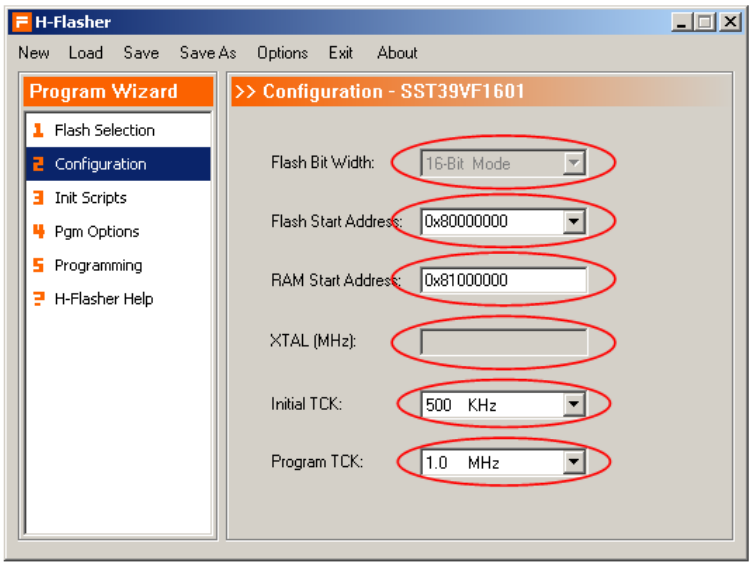

图 5-10 配置设置

5.5.3 Init Script

根据 LPC2210 的数据手册，我们需要对三个寄存器进行设置：PINSEL2@0xE002C014，BCFG0 @0xFFE00000 和 BCFG1@0xFFE00004。其中 PINSEL2 是管脚选择控制寄存器，用以确定复用管脚的功能，例如地址线和读写控制信号。而 BCFGx 则是对 BANK0 和 BANK1 进行配置，用以设置读写等待周期，数据位宽等。寄存器的具体定义，请参考 LPC2210 的数据手册。在本例中，我们需要 3 条初始化脚本，最终的脚本设置如下图所示。

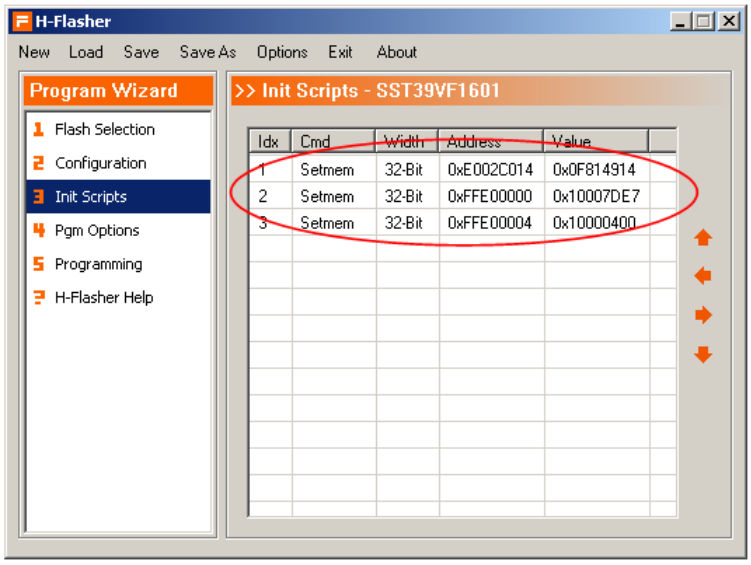

图 5-11 设置初始化脚本

5.5.4 Program Options

在向导的第四步中，用户可以选择不同的选项。在这里，我们选择了在烧写后执行系统复位，并执行二次验证，如下图所示：

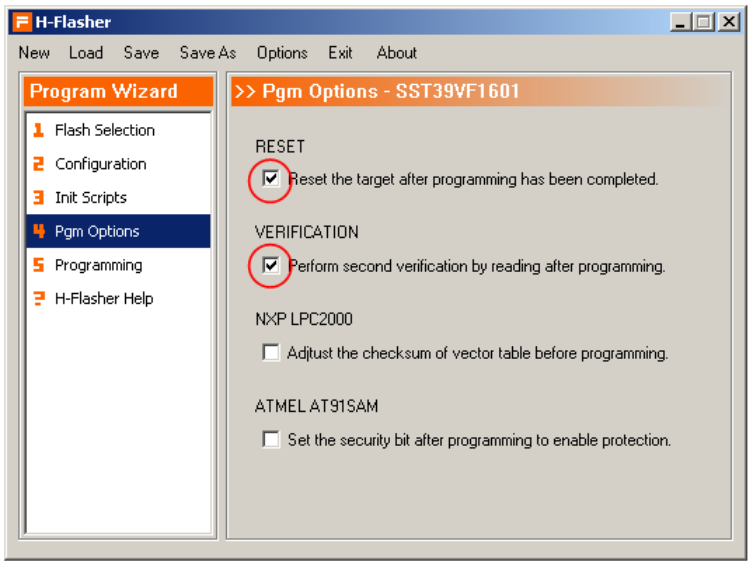

Fig 5-12 编程选项

5.5.5 Programming

在配置好后，在编程向导的第五步中，就可以对 FLASH 执行不同的操作了。首先，可以试一试 CHECK 操作。通过 CHECK 操作，用户可以大概判断前面的设置是否正确。在本例中，CHECK 的结果如下图所示。由图中显示的信息可以判断，前面的设置是正确的。

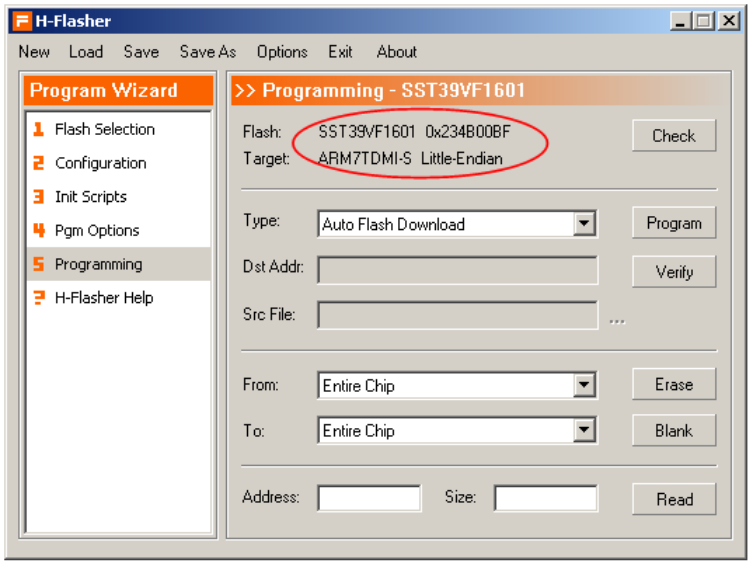

图 5-13 CHECK 操作结果

下面尝试烧写一个二进制文件。在本例中，选择文件格式为二进制，源文件为 C:\TEST.bin，烧写的目的地址为 0x80000000，设置如图 5-14 所示。设置好后，点击 PROGRAM 按钮开始烧写。烧写过程当中，用户会看到烧写文件的大小，平均烧写速度，当前的进度等信息。烧写完成后，H-FLASHER 会提示烧写并验证成功，如果图 5-15 所示。

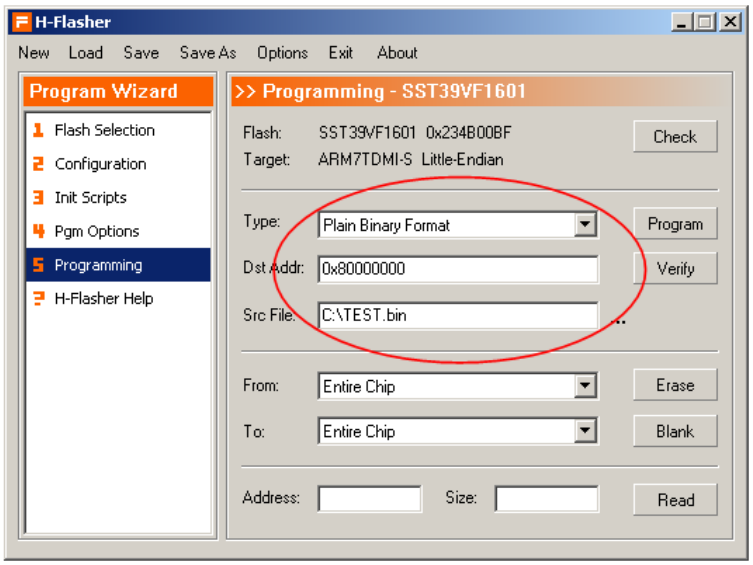

图 5-14 烧写设置

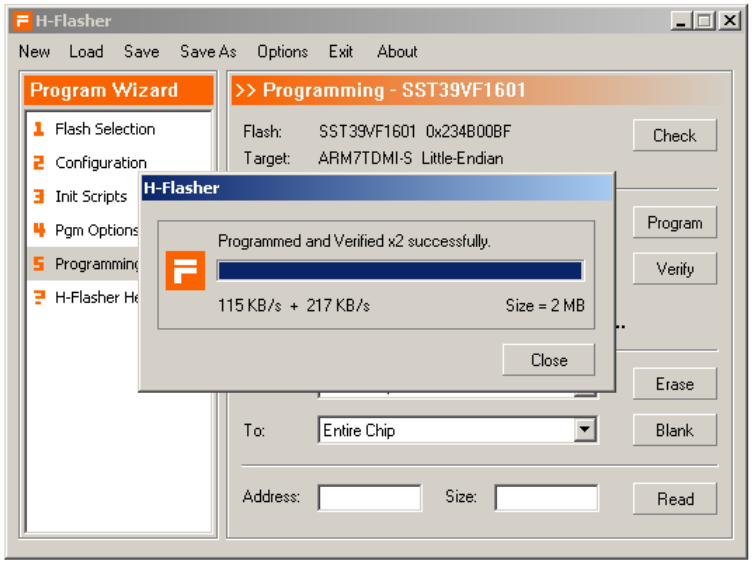

图 5-15 烧写完成

5.5.4 保存设置

如前面的例子一样，用户可以将上面的设置保存为 HFC 配置文件，供以后使用。在以后的使用中，可以直接在 H-FLASHER 里装载 HFC 文件。

第六章 初始化脚本

这个章节将简单介绍 H-JTAG 定义的初始化脚本，并介绍如何利用 H-JTAG/H-FLASHER 的脚本编辑器编辑初始化脚本。

6.1 初始化脚本的定义

H-JTAG 总共定义了 4 条初始化脚本：*Setmem*，*Getmem*, *Delay* 和 *SoftReset*，其作用分别是设置内存的值，读取内存的值, 增加延迟和执行软复位，如表 6-1 所示。用户可以根据自己的需要，使用任意的脚本组合对系统进行初始化。

表 6-1 初始化脚本

初始化脚本	脚本作用
Setmem	设置内存/寄存器的值
Getmem	读取内存的值
Delay	添加延迟
SoftReset	执行软复位

提示：

H-JTAG 目前只定义了几个简单的脚本命令，这些命令可以满足大部分的需求。但 H-JTAG 定义的初始化脚本有可能会根据实际的应用的需要而扩充。

下面分别介绍这个几条初始化脚本的格式。

6.1.1 *Setmem*

Setmem 是最常用的脚本。利用这个脚本，用户可以设置内存或是寄存器的值，用以完成系统的初始化。*Setmem* 的具体格式是：

Setmem 位宽 目的地址 目标值。

- *Setmem* — 脚本命令；
- 位宽 — 表示要设置的目标值的位宽，可选的位宽为 8-Bit/16-Bit/32-Bit；
- 目的地址 — 表示的是操作的目的地址，设置的时候请根据选择的位宽保证地址是对齐的；
- 目标值 — 用户希望写到目的地址的值，设置的时候请根据选择的位宽输入合适的值；

例子：

Setmem 08-Bit 0x0 0x12 — 将地址 0x0 的值设为 0x12，位宽为 8-Bit

Setmem 16-Bit 0x0 0x1234 — 将地址 0x0 的值设为 0x1234，位宽为 16-Bit

Setmem 32-Bit 0x0 0x12345678 — 将地址 0x0 的值设为 0x12345678，位宽为 32-Bit

6.1.2 *Getmem*

Getmem 可以用来读取内存或是寄存器的值. 这个脚本命令主要是用在特定的场合, 例如用户在配置过程中需要读取某个地址的值. *Getmem* 的具体格式是：

Getmem 位宽 目的地址。

- *Getmem* — 脚本命令；
- 位宽 — 表示读取的目标值的位宽，可选的位宽为 8-Bit/16-Bit/32-Bit；
- 目的地址 — 表示的是操作的目的地址，设置的时候请根据选择的位宽保证地址是对齐的；

例子：

Getmem 08-Bit 0x0 — 读取地址 0x0 的值，位宽为 8-Bit
Getmem 16-Bit 0x0 — 读取地址 0x0 的值，位宽为 16-Bit
Getmem 32-Bit 0x0 — 读取地址 0x0 的值，位宽为 32-Bit

6.1.3 *Delay*

Delay 可以用来添加延迟。在执行初始化的时候，有时候有必要在执行完一条指令后，等待一定的时间，再执行后面的操作。这种情况下，用户可以使用 *Delay* 脚本命令。*Delay* 的具体格式是：

Delay 延迟值（单位：毫秒）

- *Delay* — 脚本命令
- 延迟值 — 延迟的时间，单位是毫秒；

例子：

Delay 100 — 延迟 100 毫秒
Delay 5000 — 延迟 5000 毫秒

6.1.4 *SoftReset*

SoftReset 可以用来执行软复位。该操作主要是用来复位 CP15 控制寄存器，用以关闭 CACHE，禁用 MMU 等。S3C2410 是三星的一款基于 ARM920T 的芯片，支持 MMU。在将 LINUX 烧录到 S3C2410 后，LINUX 会自动配置 MMU，进行复杂的 REMAP 操作。如果用户想重新烧写 FLASH，最好在初始化脚本的开头加入 *SoftReset* 命令，这样可以关闭 CACHE 和禁用 MMU，保证目标系统的存储是用户所期望的。*SoftReset* 的具体格式是：

SoftReset （没有任何参数）

6.2 初始化脚本的编辑

H-JTAG 和 H-FLASHER 都提供了脚本编辑器，方便用户编辑脚本。H-JTAG 和 H-FLASHER 的脚本编辑器界面分别如图 6-1 和图 6-2 所示。H-JTAG 和 H-FLASHER 的编辑器是一样的，在下面的介绍中，我们将不做区分。

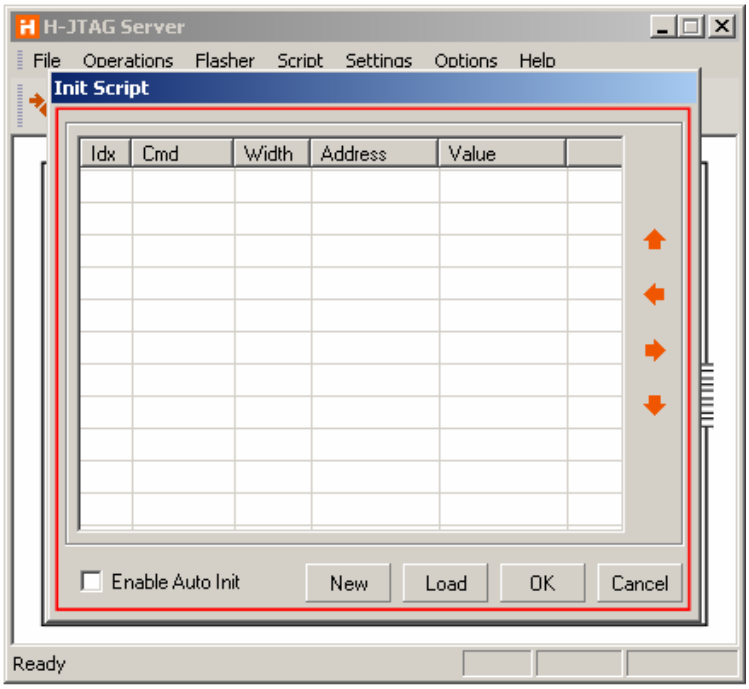

图 6-1 H-JTAG 脚本编辑器

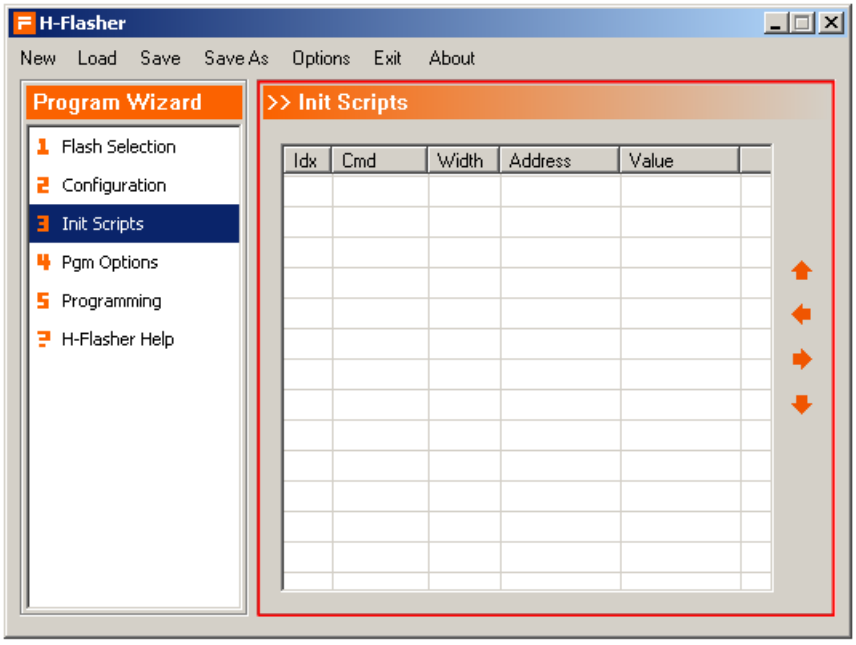

图 6-2 H-FLASHER 脚本编辑器

6.2.1 脚本编辑按钮

在脚本编辑器中，提供了 4 个箭头状的按钮，分别用来添加、删除、上移和下移脚本，具体定义如下：

将当前选择的脚本命令向上移动
添加一个新的脚本命令
删除当前选择的脚本命令
将当前选择的脚本命令向下移动

6.2.2 编辑脚本

对每条新脚本，用户需要先选择脚本命令，然后根据定义提供必要的参数。要添加一条新的脚本，在脚本编辑器中，单击右侧的“添加”按钮。添加新脚本后，如下图所示：

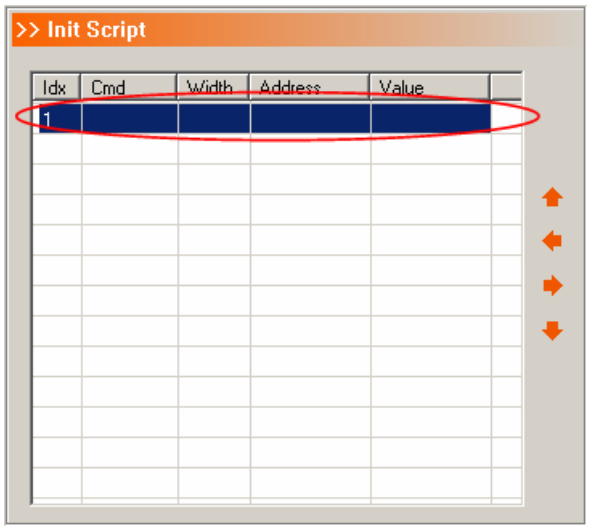

图 6-3 添加新脚本

点击脚本的 Cmd 列，用户就可以看到如下图所示的脚本命令列表。用户可以根据需要选择命令。

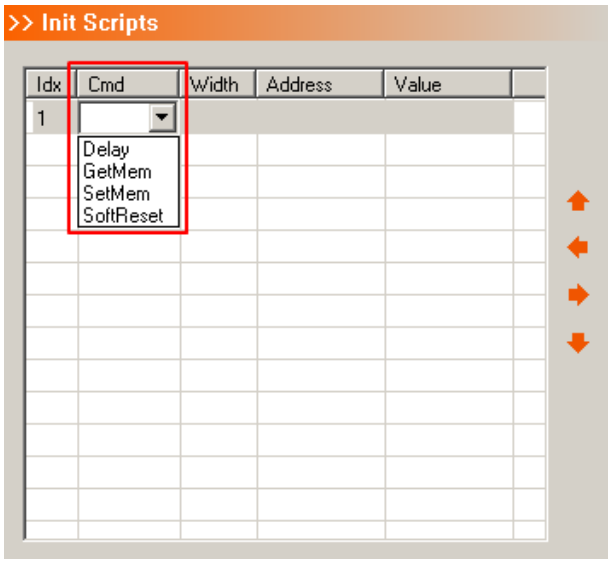

图 6-4 脚本命令列表

如果用户选择的是 *SoftReset* 命令，根据定义，用户不需要在提供别的参数。*SoftReset* 命令设置完成后，如下图所示：

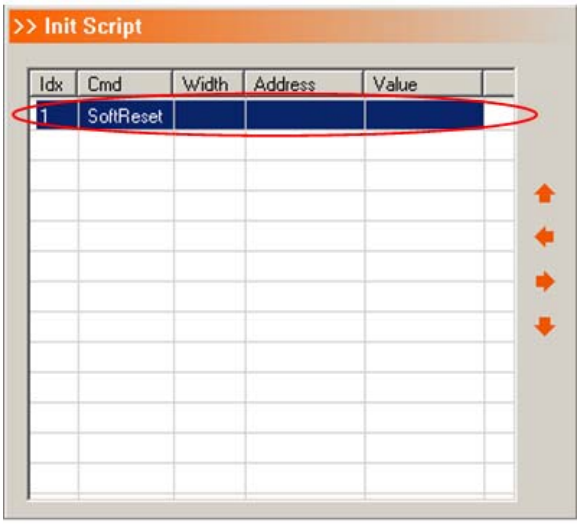

图 6-5 设置 SoftReset 脚本

如果用户选择了 *Delay* 命令，根据 *Delay* 命令的定义，用户需要在 Value 列里输入期望的延迟时间。*Delay* 命令设置完成后，如下图所示：

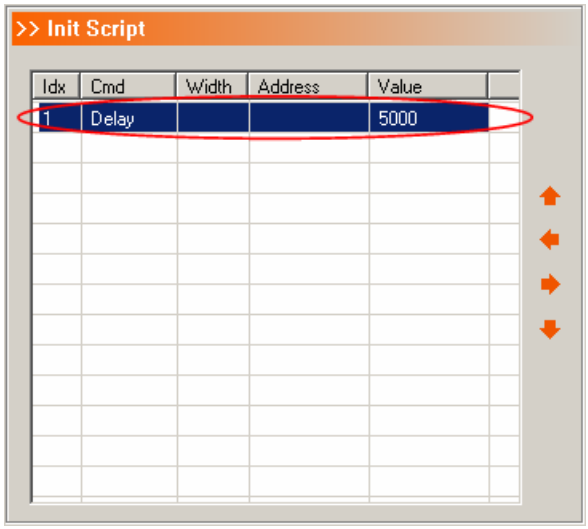

图 6-6 设置 Delay 脚本

如果用户选择了 *Setmem* 命令，根据 *Setmem* 命令的定义，用户需要设置位宽，目的地址和目标值。对于 *Setmem* 命令，点击 Width 列，可以选择位宽，如图 6-7 所示。选择好位宽后，用户需要在 Address 列和 Value 列分别填入目的地址和目标值。*Setmem* 命令设置完成后如图 6-8 所示。

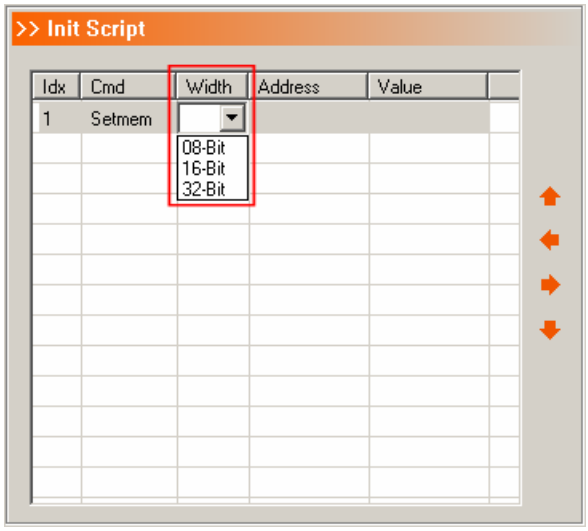

图 6-7 选择位宽

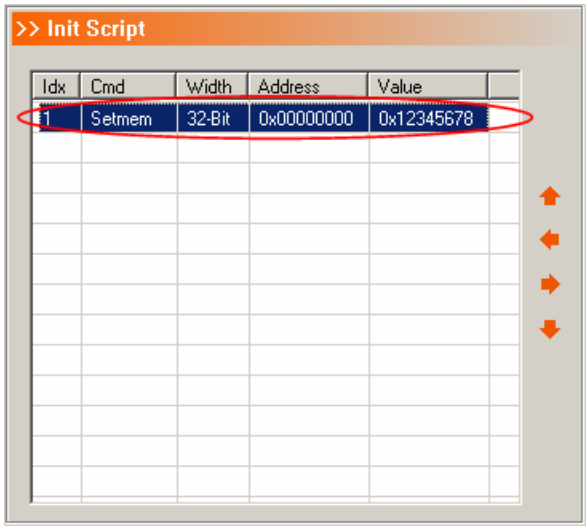

图 6-8 设置 Setmem 脚本

如果用户选择了 *Getmem* 命令，根据 *Getmem* 命令的定义，用户需要设置位宽和目的地址。对于 *Getmem* 命令，点击 Width 列，可以选择位宽，如图 6-9 所示。选择好位宽后，用户需要在 Address 列填入读操作的目的地址。*Getmem* 命令设置完成后如图 6-10 所示。

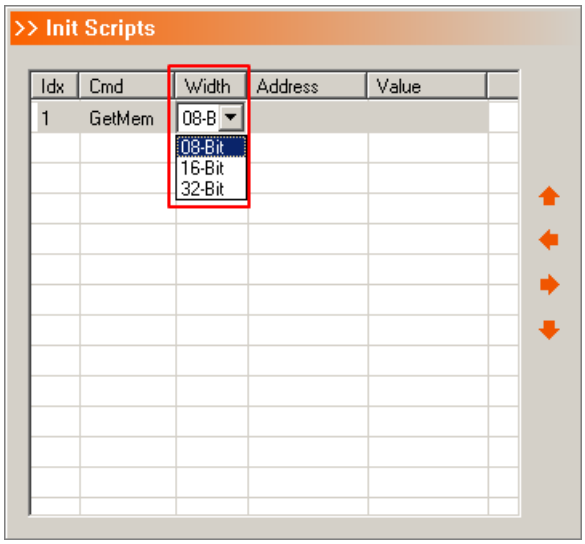

图 6-9 选择位宽

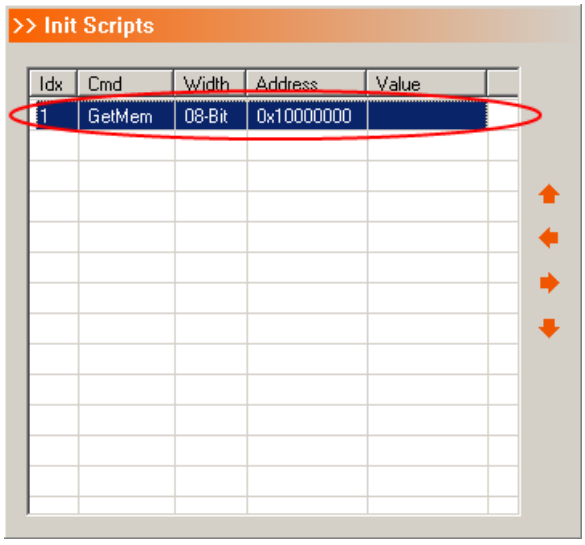

图 6-10 设置 Getmem 脚本

第七章 调试软件的配置

这个章节介绍了如何配置常用的调试软件（Debugger），以使用 H-JTAG 进行程序调试。本章介绍的调试软件包括：AXD、 RVDS、 IAR 和 KEIL。在设置完成后，要确保调试的顺利进行，请正确连接好硬件，打开 H-JTAG，并检测到调试目标。调试软件的具体使用，请参考各个软件自带的用户手册。

7.1 AXD 的配置

ADS 的全称是 ARM DEVELOPER SUIT，是 ARM 公司原厂的 IDE，拥有众多的用户。而 AXD 是 ADS 自带的调试软件。下面介绍如何配置 AXD 以配合 H-JTAG 进行调试。

首先，点击 Options -> Configure Target 菜单，如图 7-1 所示。

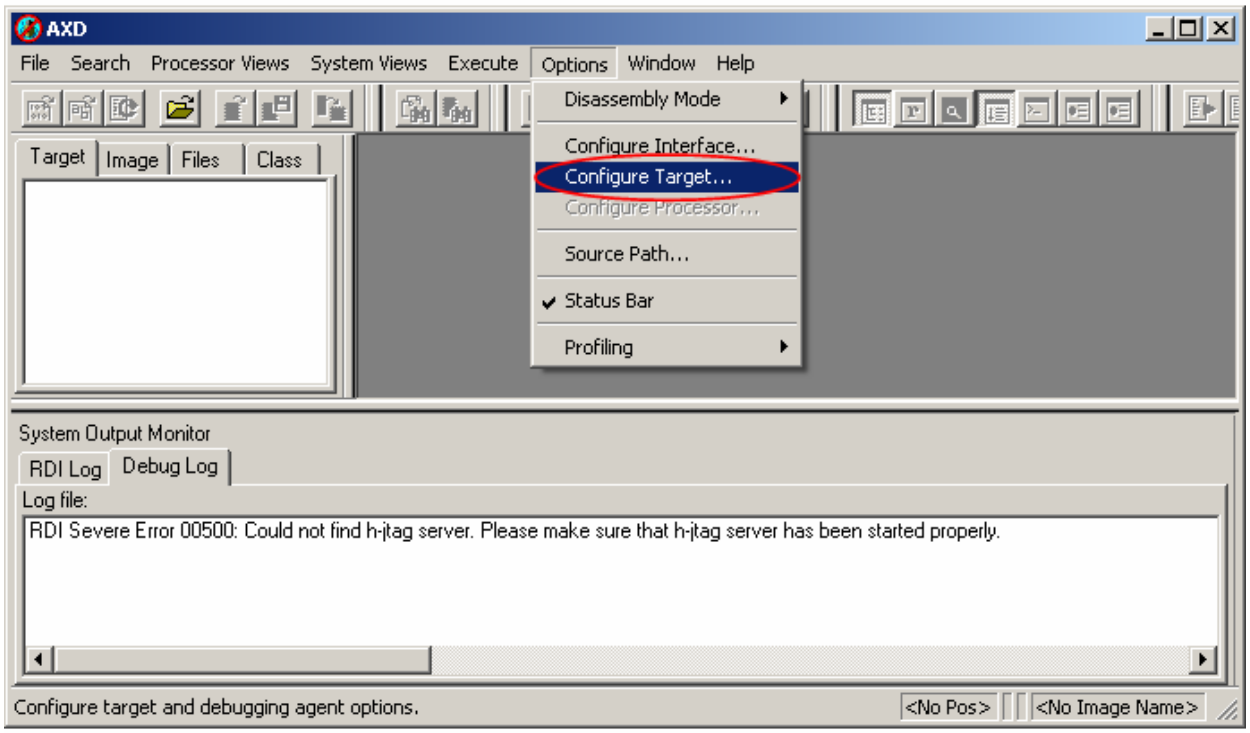

图 7-1 Configure Target 菜单

接下来，用户会看到如图 7-2 所示的 Choose Target 配置窗口。

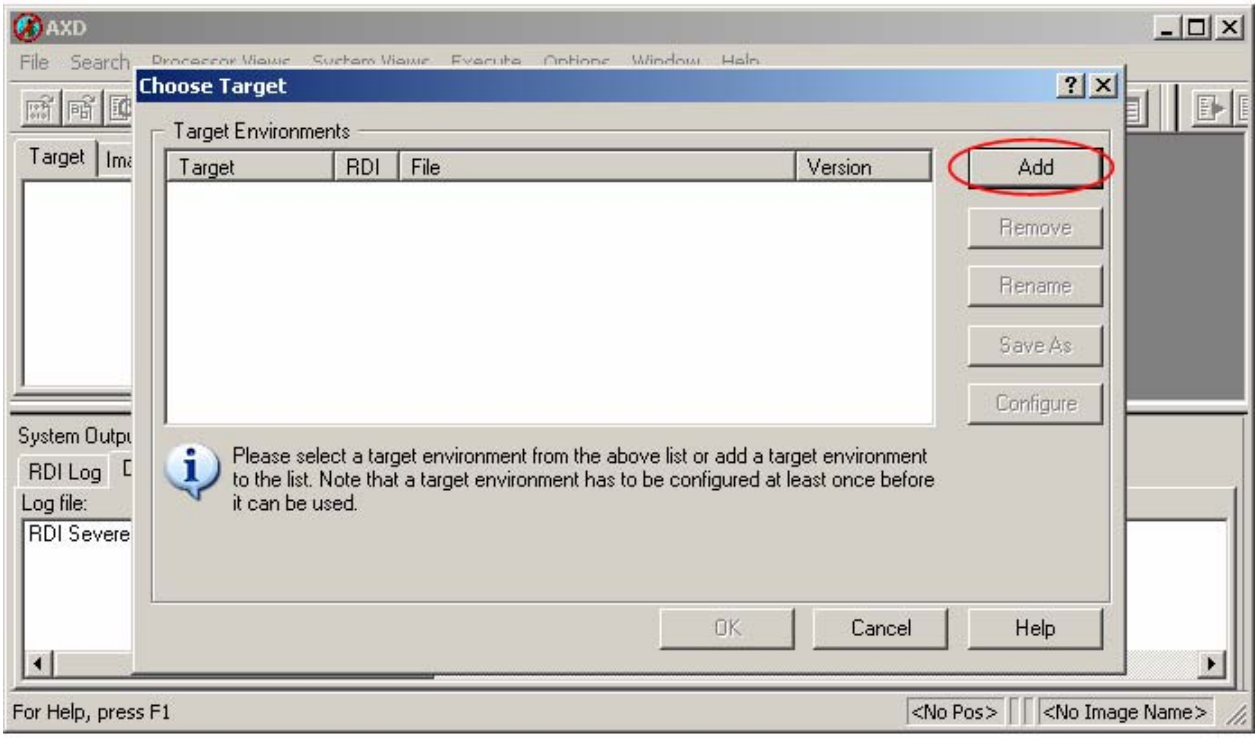

图 7-2 Choose Target 配置窗口

在上图所示的配置窗口中，点击 Add 按钮，用户会看到如图 7-3 所示的选择 DLL 文件的对话框。在对话框里选择 H-JTAG 安装目录下的 H-JTAG.DLL，然后点击确定。

图 7-3 选择 H-JTAG.DLL

确定后，用户可以看到 H-JTAG 已经添加到 AXD 里去了，如图 7-4 所示。在下图中，双击 H-JTAG 或是单击 Configure 按钮，用户就可以看到如图 7-5 所示的 H-JTAG 信息。在图 7-4 中，点击 OK，AXD 的配置就全部完成了。

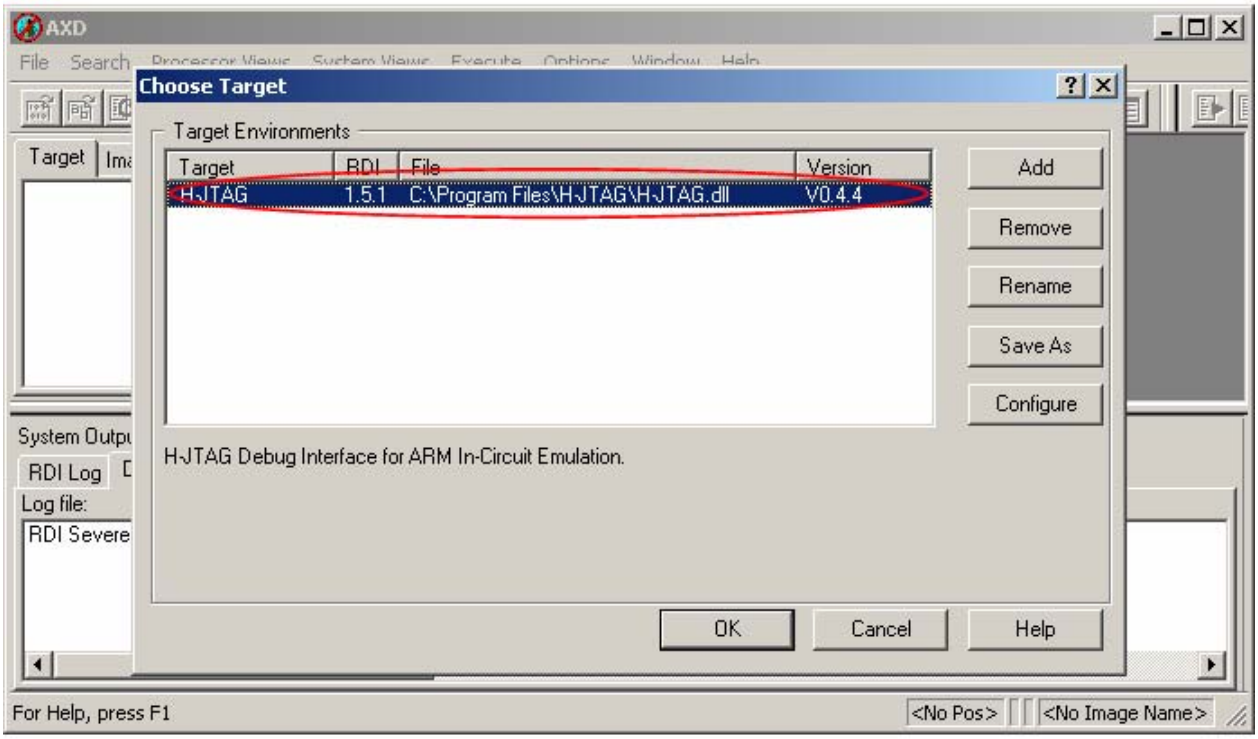

图 7-4 配置完成

图 7-5 H-JTAG 信息

7.2 RVDS 的配置

RVDS 的全称是 REALVIEW DEVELOPER SUIT，也是 ARM 公司原厂的 IDE。下面将以 RVDS2.0 为例，介绍如何配置 RVDS 以配合 H-JTAG 进行调试。RVDS2.2 的配置稍有不同，但基本一样。

首先，点击 RVDS 主界面上的“Click to connect a target”，如下图所示：

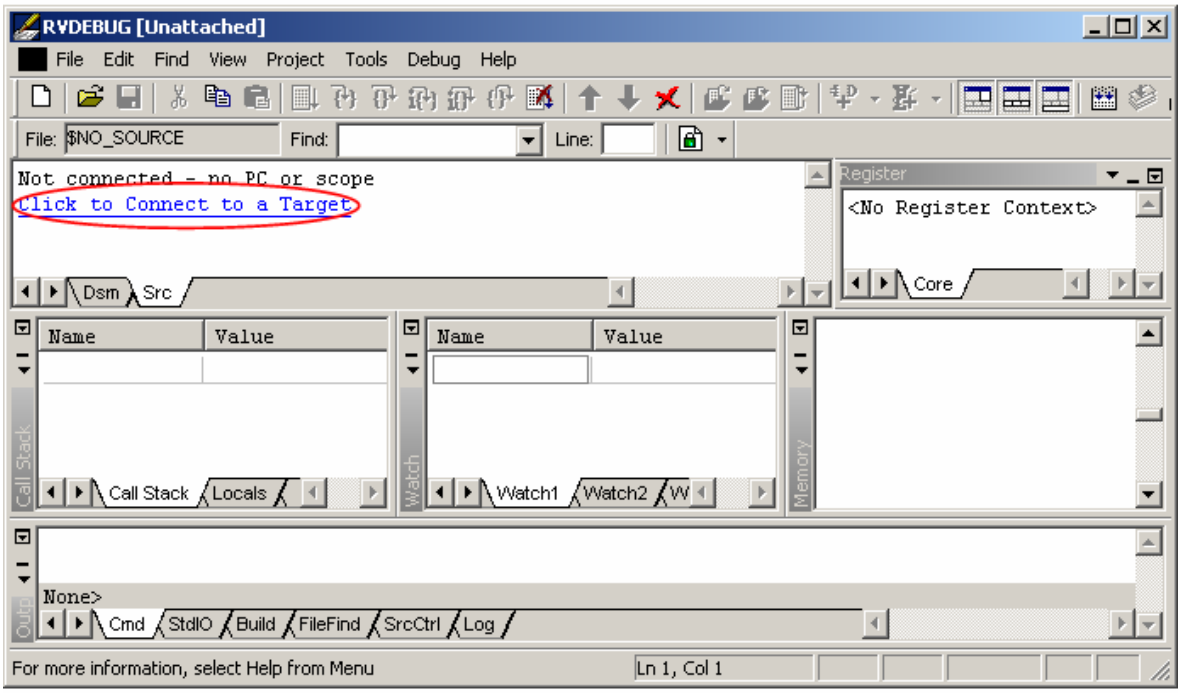

图 7-6 RVDS 主窗口

接下来，用户会看到如图 7-7 所示的 Connection Control 配置窗口。

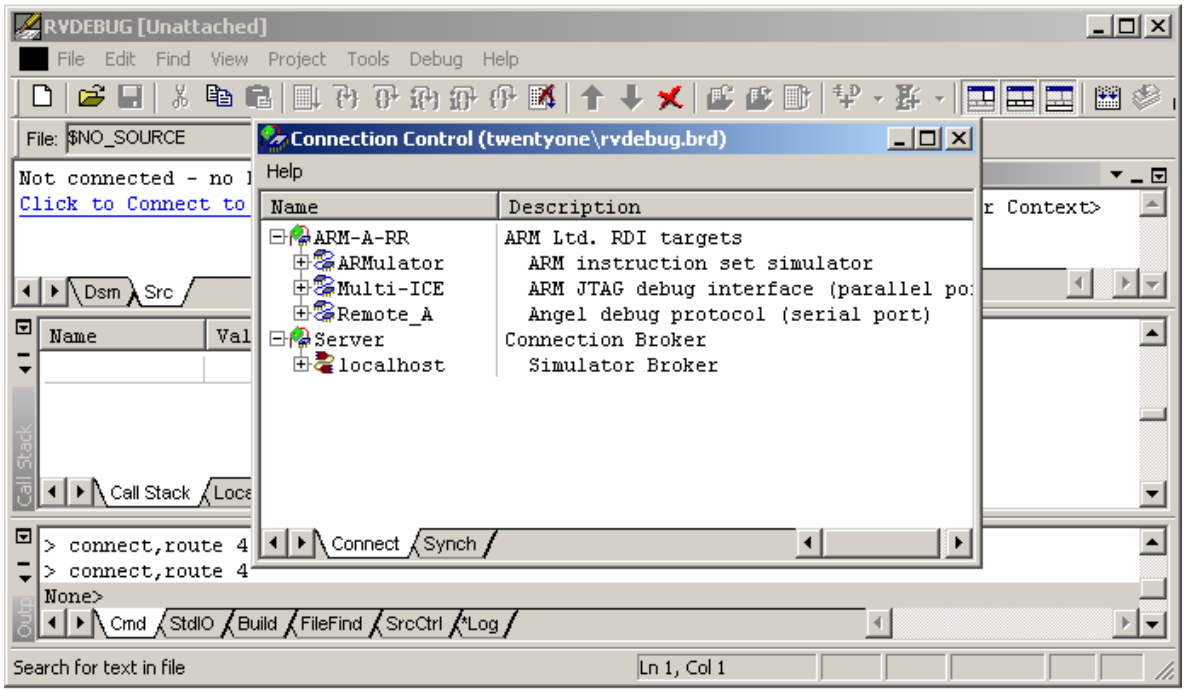

图 7-7 Connection Control 配置窗口

在上图的配置窗口中，单击右键，就可以看到如图 7-8 所示的菜单。

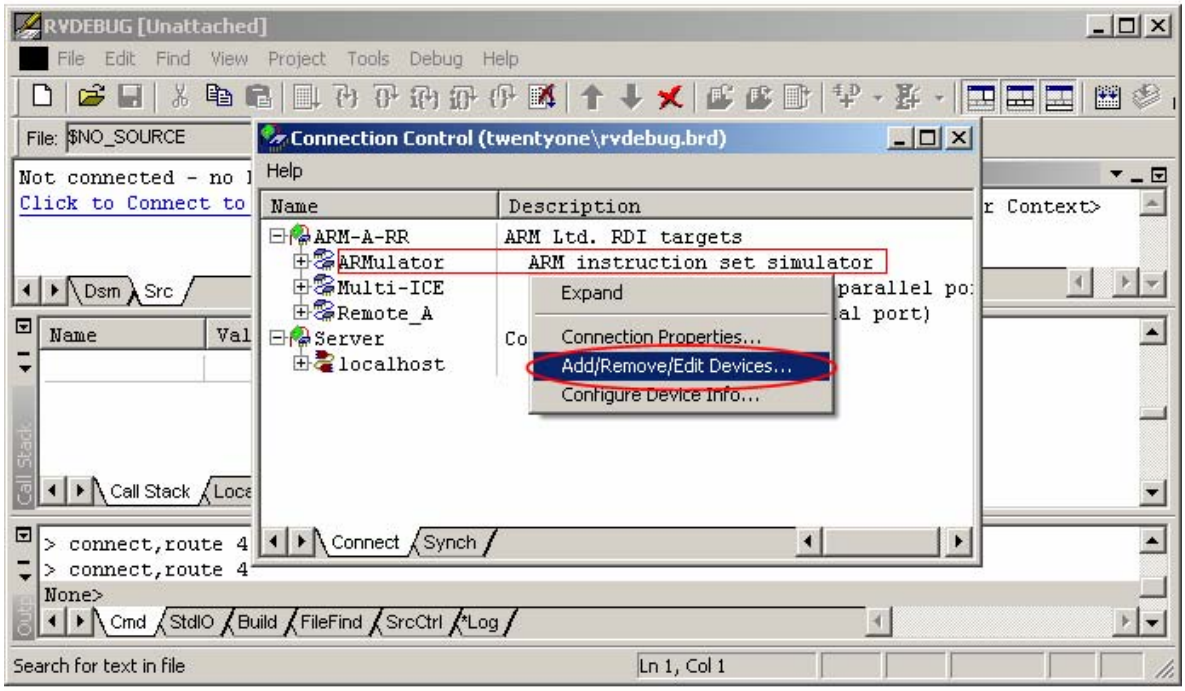

图 7-8 添加菜单

在图 7-8 所示的菜单中，选择 Add/Remove/Edit Devices，就可以看到如下图所示的 RDI Target List 窗口。

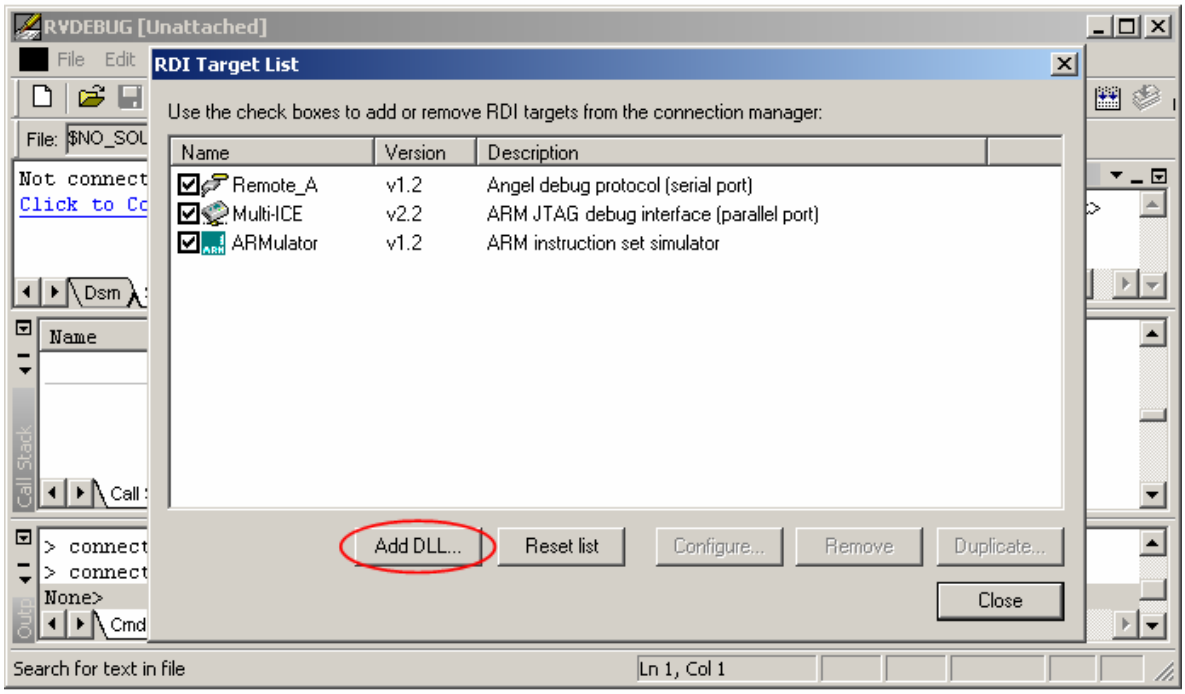

图 7-9 RDI Target List 窗口

在图 7-9 中，选择 Add DLL，用户会看到如 7-10 所示的 Select RDI DLL 文件的窗口。在这个窗口中，选择 H-JTAG 安装目录下的 H-JTAG.DLL，然后点击确定。

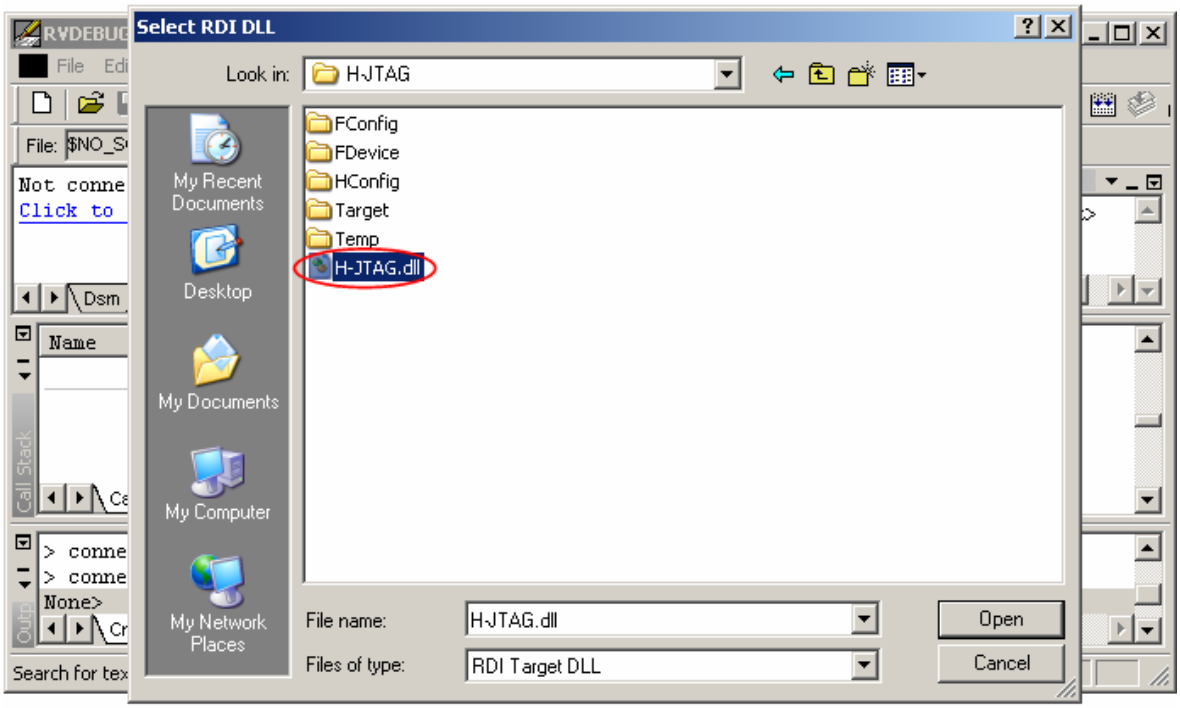

图 7-10 选择 H-JTAG.DLL

确定后，用户会看到如图 7-11 所示的 Create New RDI Target 窗口。在这个窗口中，用户可以输入名字和简单的描述，或是采用默认值。

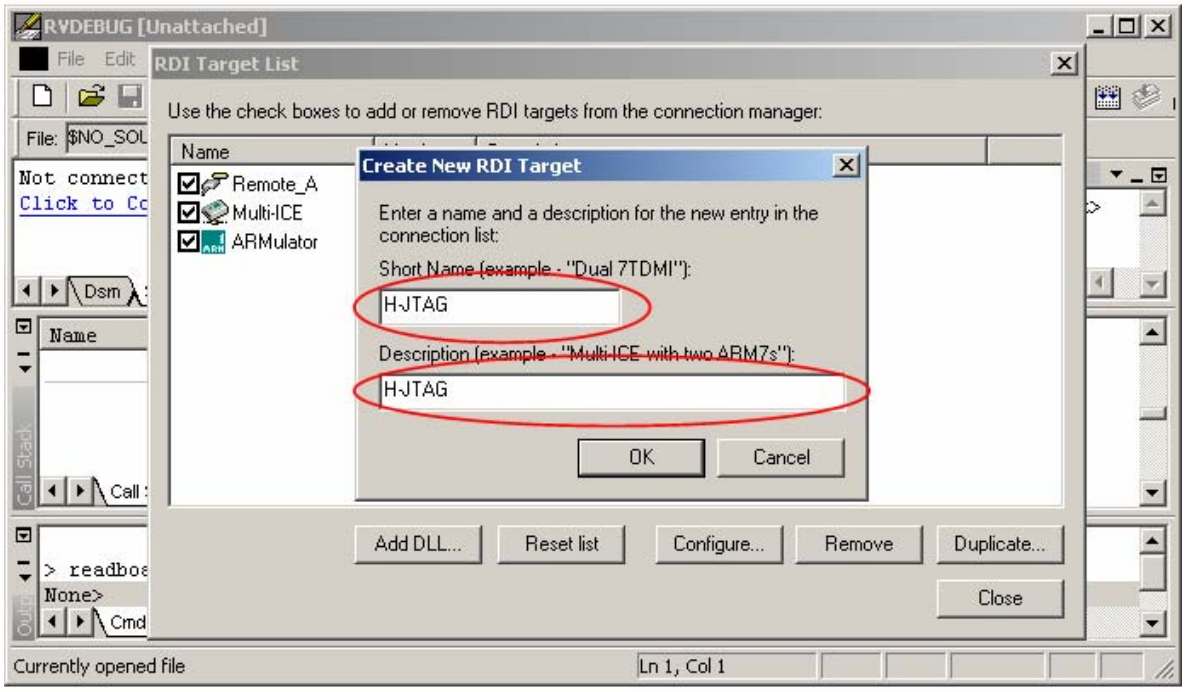

图 7-11 Create New RDI Target 窗口

确定后，用户可以看到 H-JTAG 已经添加到 RVDS 里去了，如图 7-12 所示。在下图中，单击 Configure 按钮，用户就可以看到如图 7-13 所示的 H-JTAG 信息。在图 7-12 中，点击 Close，RVDS 的配置就全部完成了。

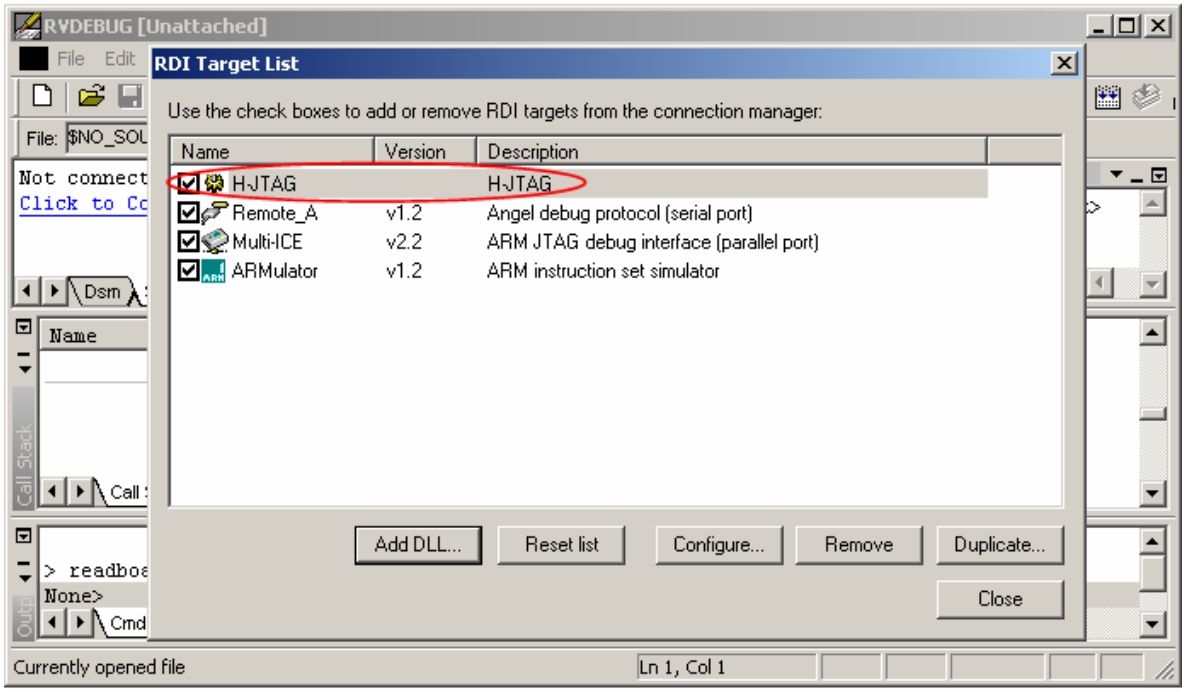

图 7-12 配置完成

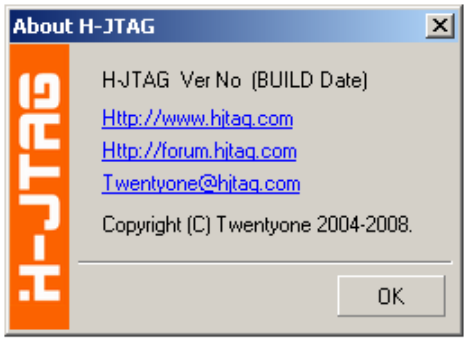

图 7-13 H-JTAG 信息

7.3 IAR 的配置

IAR Embedded Workbench 是 IAR 公司出的一款针对 ARM 处理器的 IDE。下面将介绍如何配置 IAR 以配合 H-JTAG 进行调试。

首先，在 IAR 中打开一个项目，然后点击 Project->Options 菜单，如下图所示：

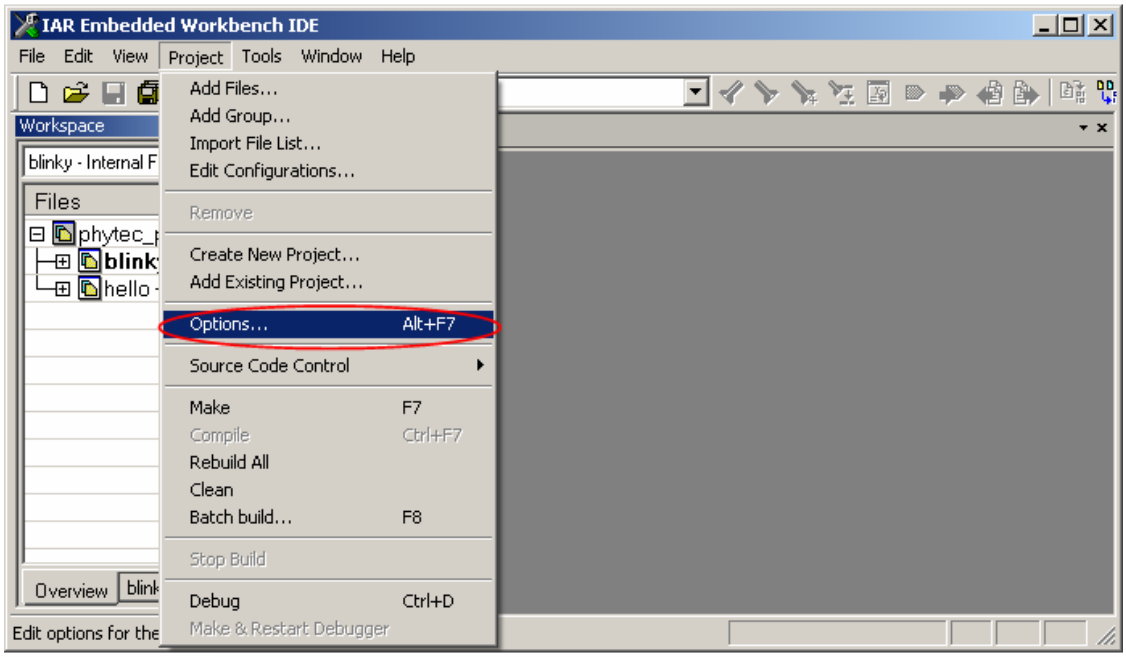

图 7-14 IAR Options 菜单

接下来，用户会看到如图 7-15 所示的 Options 配置窗口。

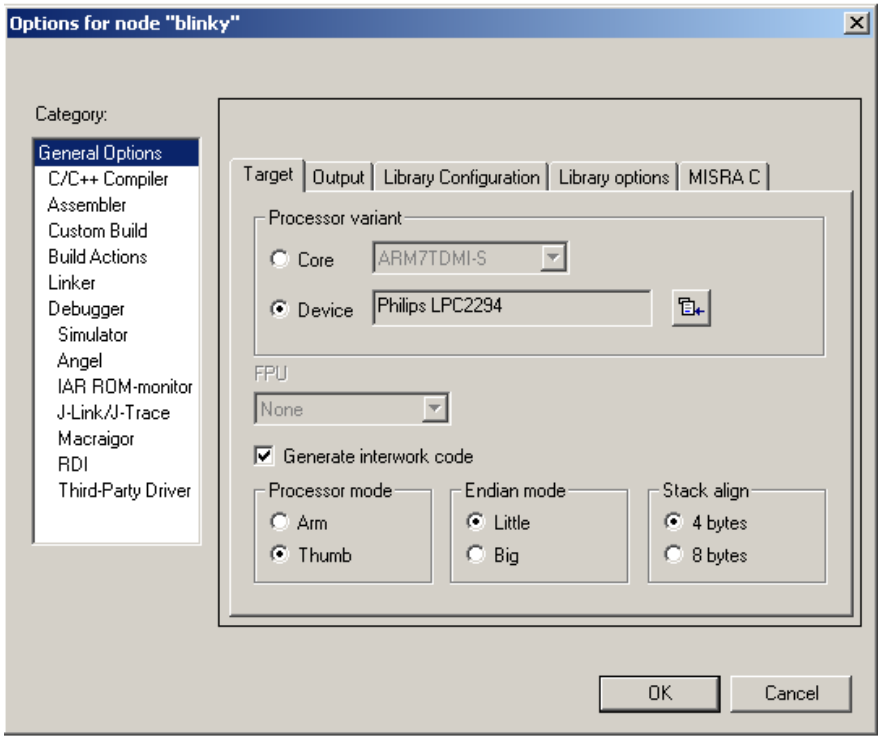

图 7-15 Options 配置窗口

在图 7-15 所示的 Options 配置窗口中，选择 Debugger 设置，并在 Setup 页面中，驱动选项中选择 RDI。如下图所示：

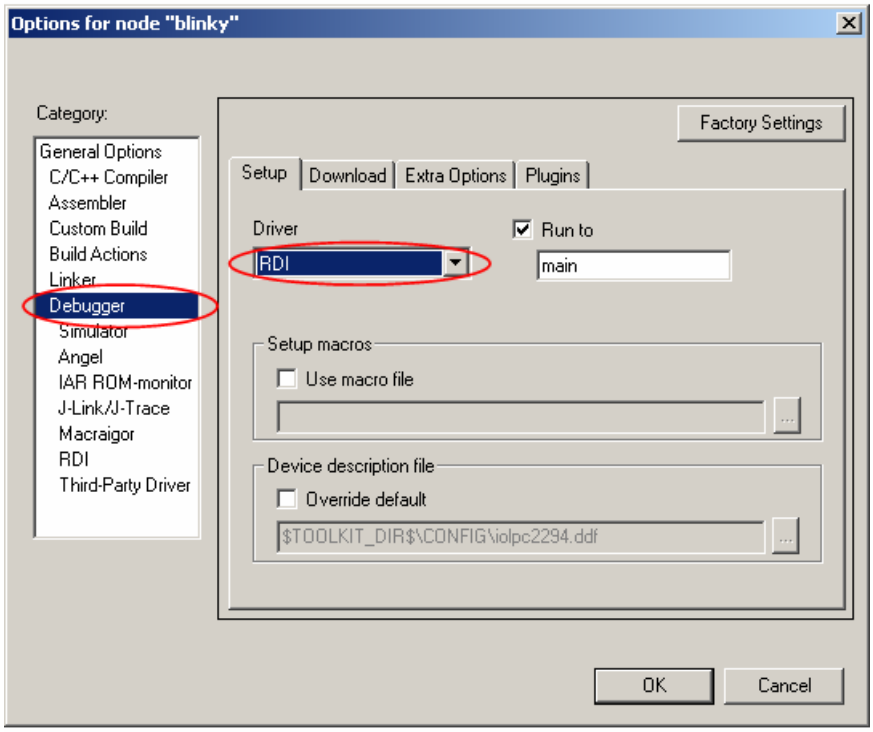

图 7-16 Debugger 设置

然后再选择 RDI 设置，如下图所示。在这个页面里，用户需要指定 RDI 驱动的路径。

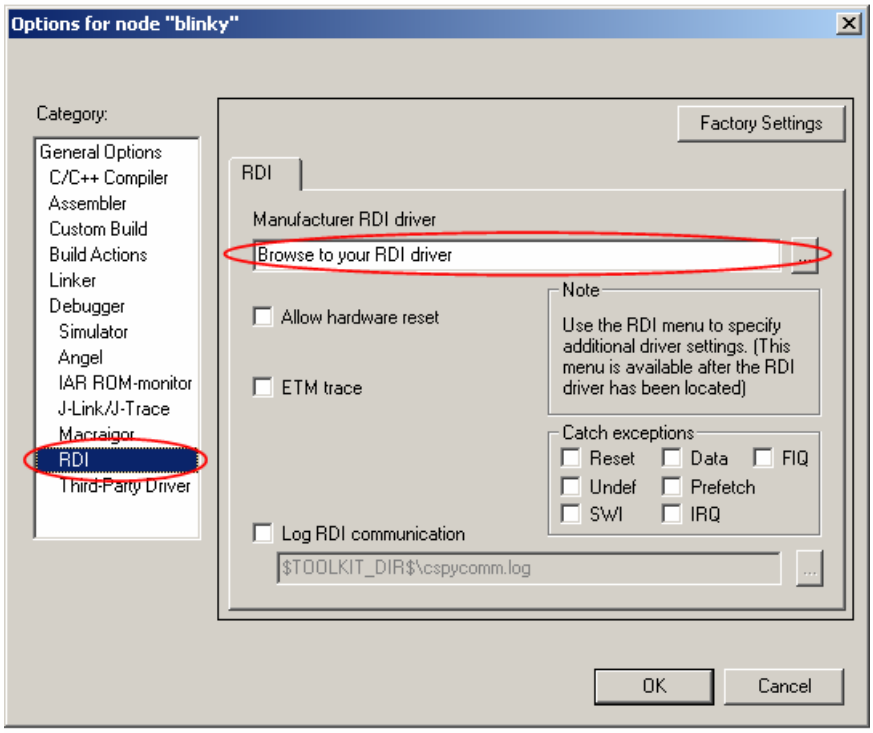

图 7-17 RDI 设置

在上图中，点击 Browse 按钮，选择 H-JTAG 安装目录下的 H-JTAG.DLL，如下图所示：

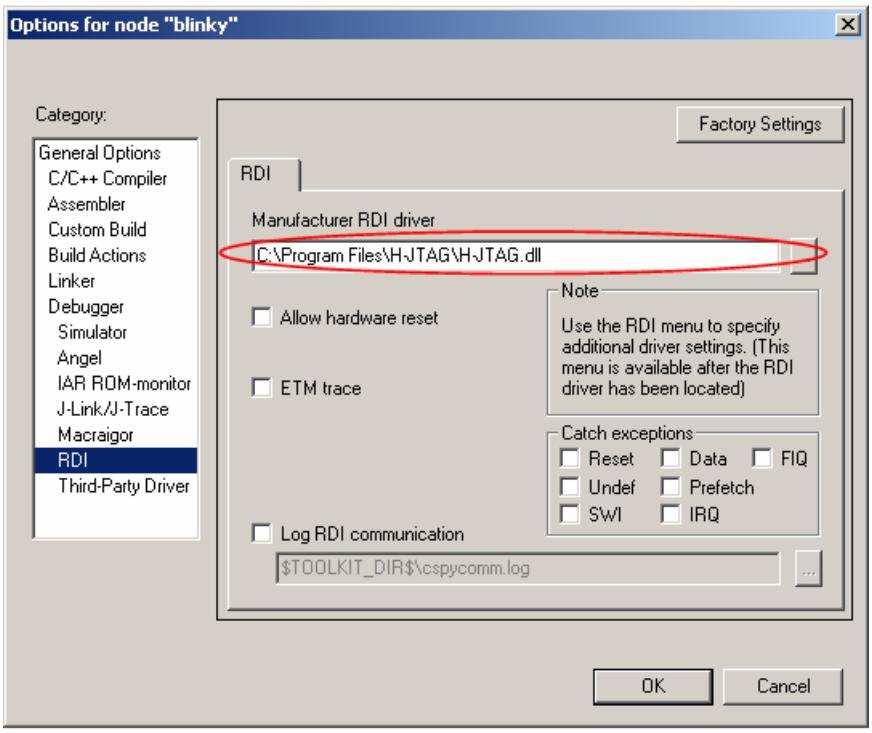

图 7-18 选择 H-JTAG.DLL

在上图中，点击 OK 按钮，IAR 的设置就完成了。设置完成后，IAR 的主窗口中多了一个 RDI 菜单，如图 7-19 所示。在菜单中点击 Configure，就可以看到如图 7-20 所示的 H-JTAG 相关信息。

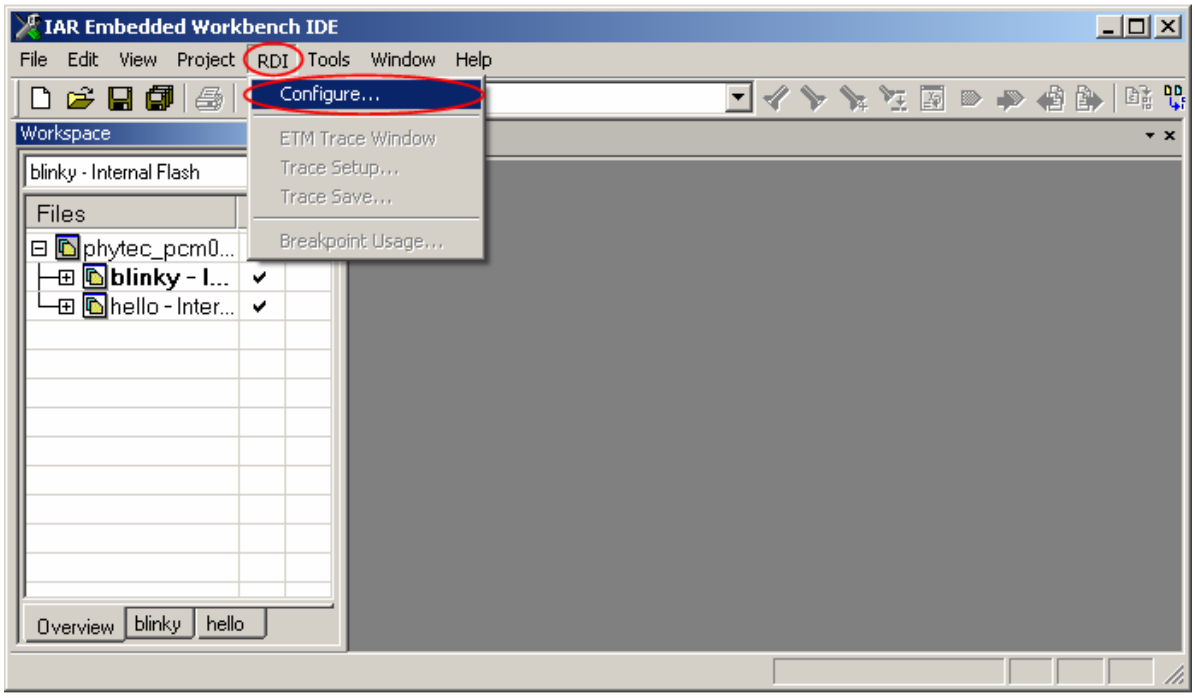

图 7-19 IAR 设置完成

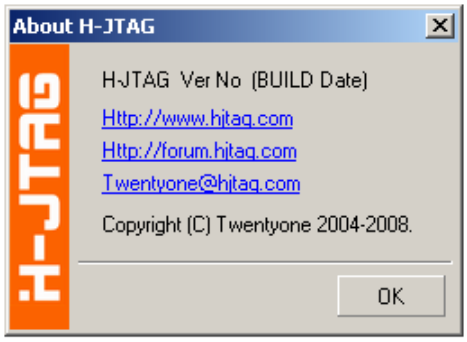

图 7-20 H-JTAG 信息

提示：

如果用户用 IAR 调试的时候希望使用 FLASH 自动下载功能，请在 IAR 的 Options 中，请不要打开 Verify Download 和 Use Flash Loader(s)选项。设置如下图所示：

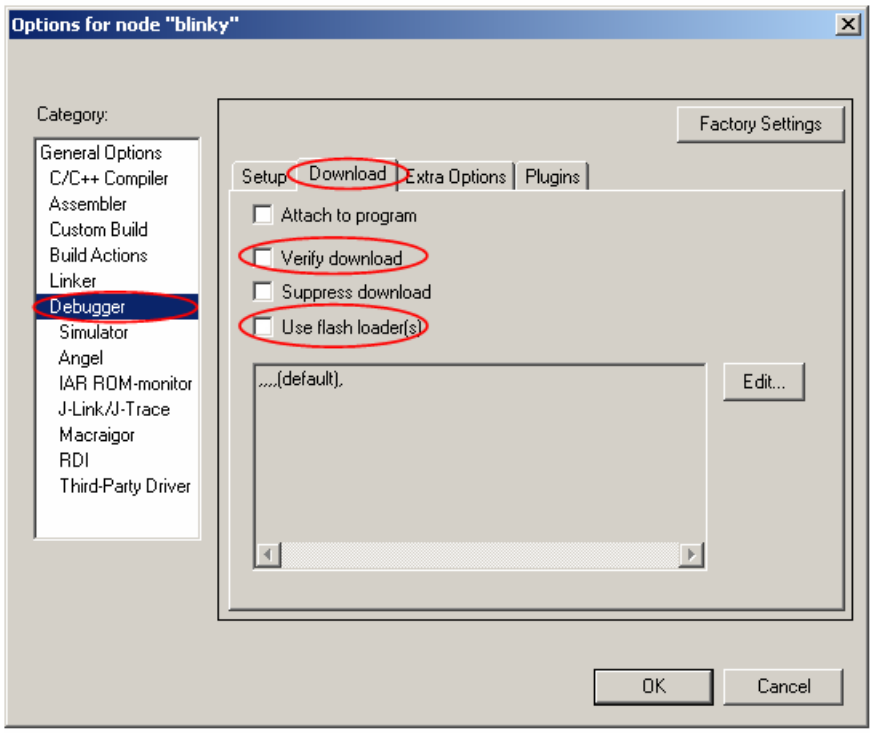

图 7-21 Verify Download 设置

7.4 KEIL 的配置

KEIL FOR ARM 是 KEIL 公司开发的 IDE。下面将介绍如何配置 KEIL FOR ARM 以配合 H-JTAG 进行调试。

首先，在 KEIL 中打开一个项目，然后点击 Project -> Options for Target…菜单，如下图所示：

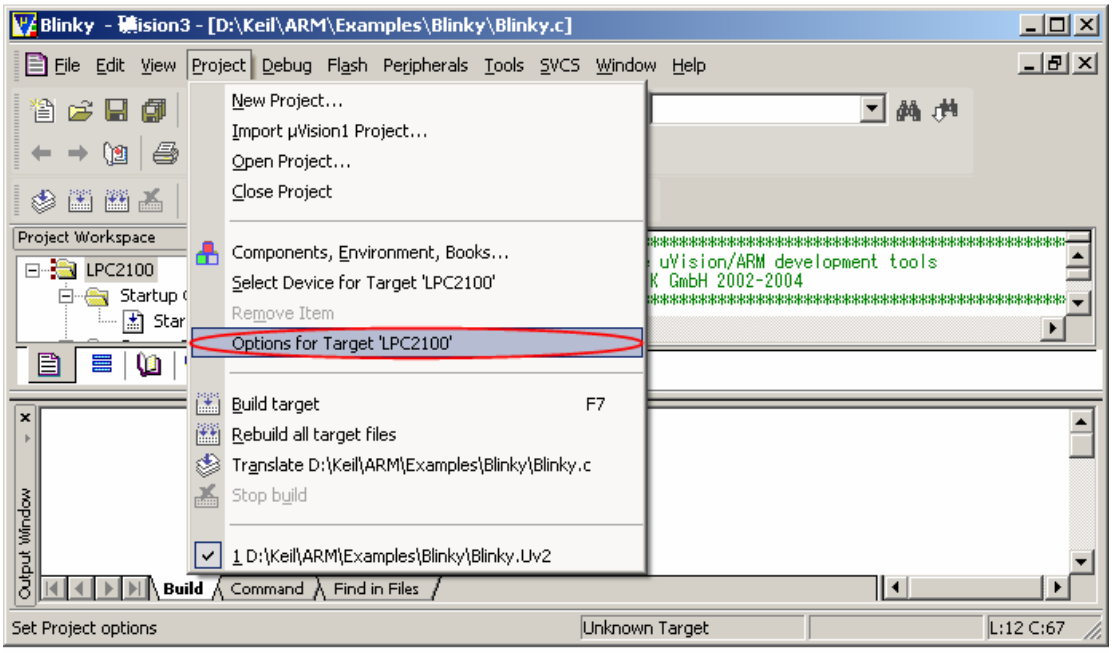

图 7-22 Options 菜单

接下来，用户会看到如图 7-23 所示的 Options 配置窗口。

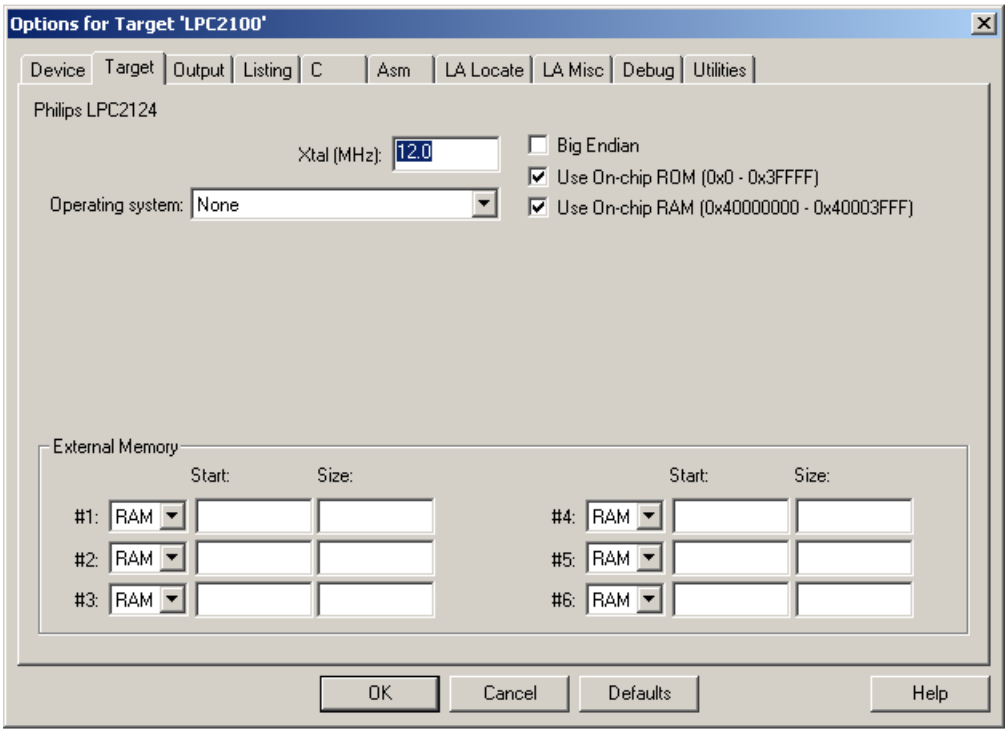

图 7-23 Options 配置窗口

在图 7-23 所示的 Options 配置窗口中，选择 Debug 设置，如下图所示：

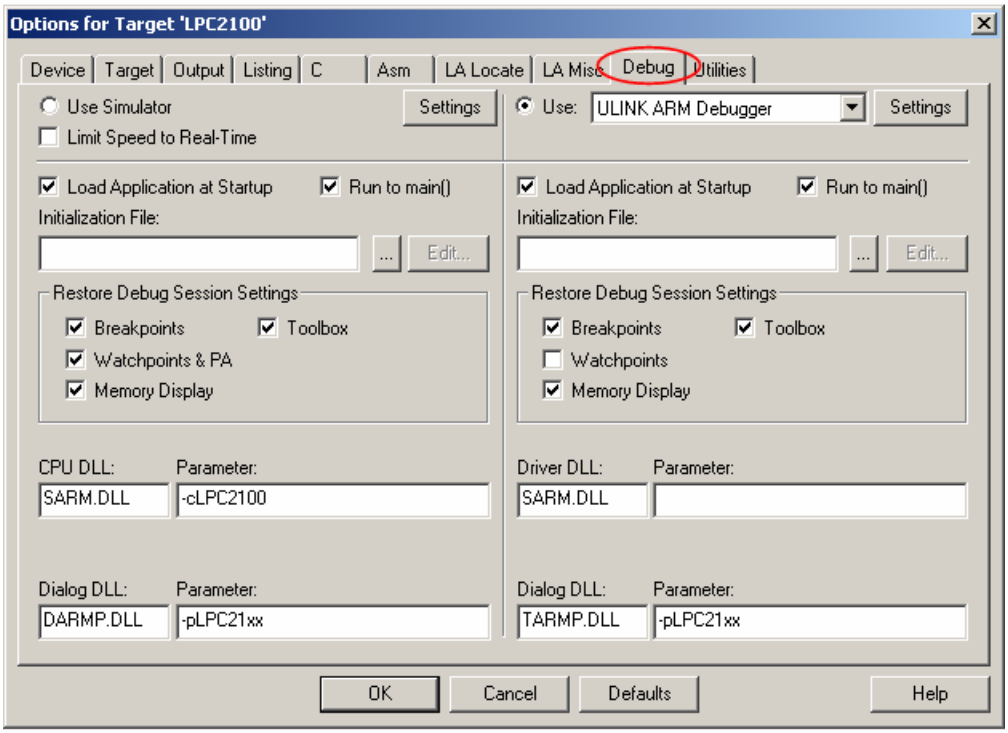

图 7-24 Debug 设置

在 Debug 设置中，选择使用 RDI 接口，如下图所示。

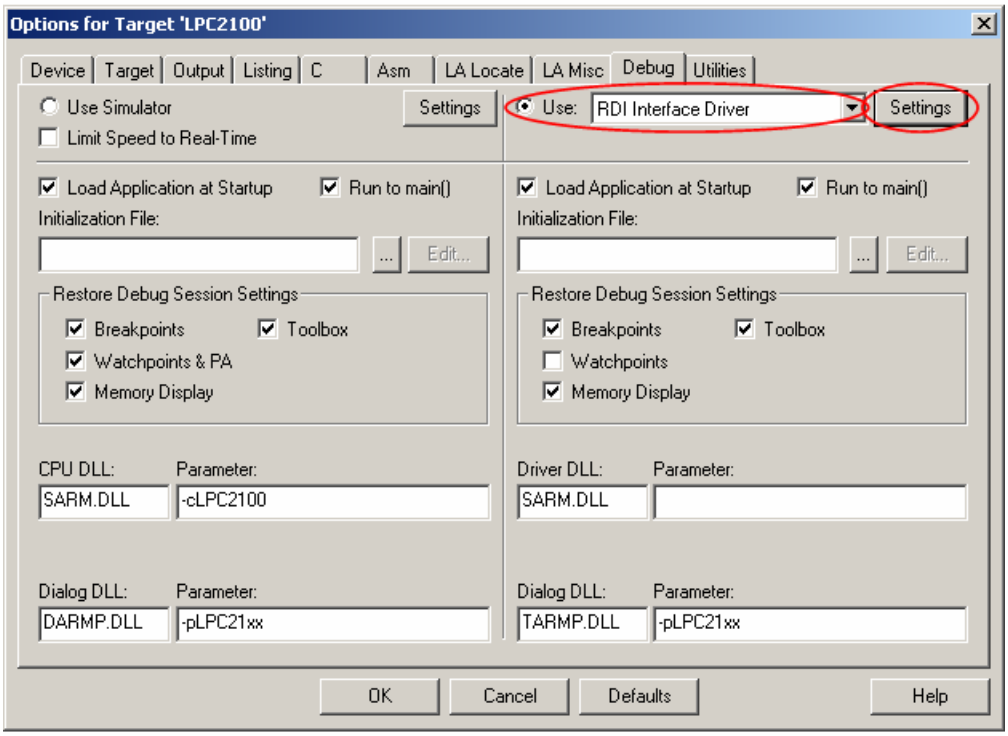

图 7-25 选择使用 RDI 接口

然后点击上图所示的 Settings 按钮，KEIL 会弹出如下图所示的 RDI Interface Driver Setup 窗口，需要用户设置 H-JTAG.DLL 的路径。

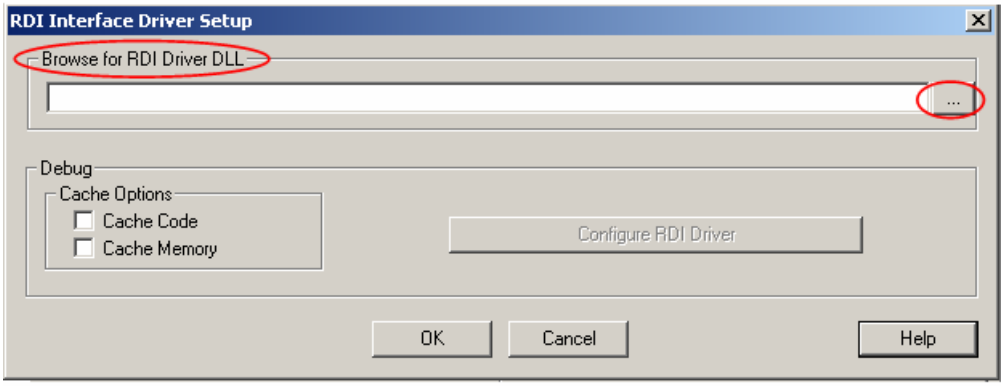

图 7-26 RDI Interface Driver Setup 窗口

在上图所示的设置窗口中，按 Browse 按钮，然后选择 H-JTAG 安装目录下的 H-JTAG.DLL，设置好路径，如下图所示。

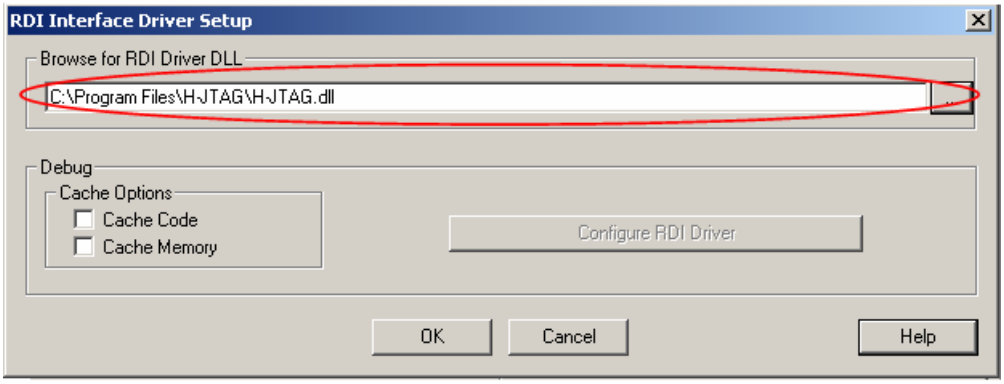

图 7-27 设置 H-JTAG.DLL 路径

提示：

在设置 H-JTAG.DLL 路径的时候，如果 H-JTAG Server 没有打开，KEIL 会报错，用户可以忽略这个错误。

在上图中，用户点击 OK 按钮，就会回到图 7-25 所示的设置窗口。继续点击 OK 按钮，回到 KEIL 的主窗口，KEIL 的设置就完成了。