华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《ARM 嵌入式 C 编程标准教程》

作者：华清远见

专业始于专注 卓识源于远见

第 2 章 ADS 1.2 开发环境创建与简介

专业始于专注 卓识源于远见

2.1 ADS 1.2 开发环境创建

2.1.1 ADS 1.2 概述

ADS 是个集成开发环境，主要包括编译器、链接器、调试器、C 和 C++库等，是 ARM 公司推出的新一代 ARM 集成开发工具。最新版本是 ADS 1.2，该版本支持包括 Windows 和 Linux 在内的多种操作环境。ADS 1.2 的组成如下所述。

1．编译器

ADS 提供多种编译器，以支持 ARM 和 Thumb 指令的编译，主要有如下几种。

- armcc：是 ARM C 编译器。
- tcc：是 Thumb C 编译器。
- armcpp：是 ARM C++编译器。
- tcpp：是 Thumb C++编译器。
- armasm：是 ARM 和 Thumb 的汇编语言编译器。

2．链接器

armlink 是 ARM 链接器。该命令既可以将编译得到的一个或多个目标文件和相关的一个或多个库文件进行链接，生成一个可执行文件，也可以将多个目标文件部分链接成一个目标文件，以供进一步的链接。

3．符号调试器

armsd 是 ARM 和 Thumb 的符号调试器，能进行源码级程序调试。用户可以在用 C 或汇编语言写的代码中进行单步调试、设置断点、查看变量值和内存单元的内容。

4．fromELF

将 ELF 格式的文件转换为各种格式的输出文件，包括 BIN 格式映像文件、Motorola 32 位 S 格式映像文件、Intel 32 位格式映像文件和 Verilog 十六进制文件。FromELF 命令也能够为输入映像文件产生文本信息，例如，代码和数据长度。

5．armar

armar 是 ARM 库函数生成器，它将一系列 ELF 格式的目标文件以库函数的形式集合在一起。用户可以把一个库传递给一个链接器以代替几个 ELF 文件。

6．CodeWarrior

CodeWarrior 集成开发环境（IDE）为管理和开发项目提供了简单、多样化的图形用户界面，用户可以使用 ADS 的 CodeWarrior 集成开发环境为 ARM 和 Thumb 处理、开发用 C、C++或者 ARM 汇编语言编写的程序代码。

7．调试器

ADS 中含有 3 个调试器，即 AXD、Armsd 和 ADW/ADU。
在 ARM 体系中，可以选择多种调试方式，如 Multi-ICE（Multi-processor In-Circuit Emulator）、ARMulator 或 Angel。
Multi-ICE 是一个独立的产品，是 ARM 公司自己的 JTAG 在线仿真器，不是由 ADS 提供的。

- ARMulator 是一个 ARM 指令集仿真器，集成在 ARM 的调试器 AXD 中，提供对 ARM 处理器的指令集的仿真，为 ARM 和 Thumb 提供精确的模拟。用户可以在硬件尚未做好的情况下开发程序代码，利用模拟器方式调试。
- Angel 是 ARM 公司常驻在目标机 Flash 中的监控程序，只需通过 RS-232C 串口与 PC 主机相连，就可以对基于 ARM 架构处理器的目标机进行监控器方式的调试。

8．C 和 C++库

ADS 提供 ANSI C 库函数和 C++库函数，支持被编译的 C 和 C++代码。用户可以把 C 库中的与目标相关的函数作为自己应用程序中的一部分，重新进行代码的实现。这就为用户带来了极大的方便，用户可以针对自己的应用程序的要求，对与目标无关的库函数进行适当的裁剪。在 C 库中有很多函数是独立于其他函数的，并且与目标硬件没有任何依赖关系。对于这类函数，用户可以很容易地在汇编代码中使用。
有了这些部件，用户就可以为 ARM 系列的 RISC 处理器编写和调试自己的开发应用程序了。

2.1.2 ADS 1.2 的安装

ADS 全称为 ARM Developer Suite，是 ARM 公司推出的新的一代 ARM 集成开发工具。现在 ADS 的最新版本是 1.2，它取代了早期的 ADS 1.1 和 ADS 1.0，该版本支持包括 Windows 和 Linux 在内的多种操作系统。安装步骤如下。
在 ADS1.2 的安装盘中运行 setup.exe，安装 ARM Developer Suite v1.2。出现如图 2-1 所示的对话框，同意产权协议，选省缺安装路径（C:\Program Files\ARM\vADS1.2）和典型安装模式（Typiflcation），单击“Next”按钮进入下一步，出现选文件夹、编程语言和当前设定对话框，如图 2-2 所示，单击“Next”按钮，开始安装，如图 2-3 所示。

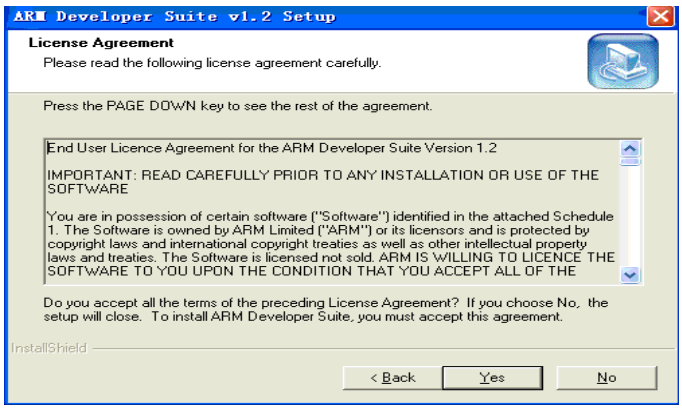

图 2-1　同意产权协议

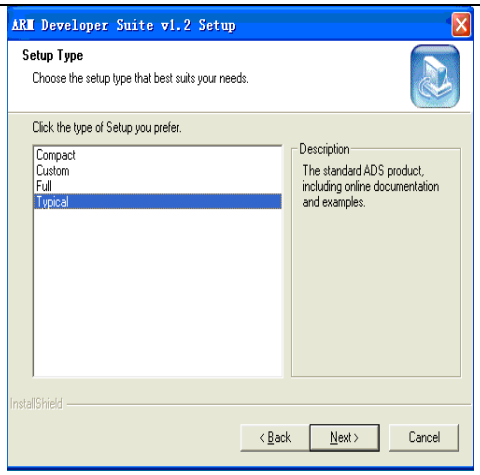

图 2-2　选典型安装模式

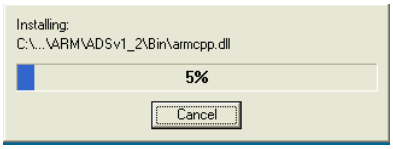

图 2-3　开始安装

安装结束，安装许可文件（Install License），这一步可按安装向导进行，单击“下一步”按钮，会出现如图 2-4 和图 2-5 所示的对话框。

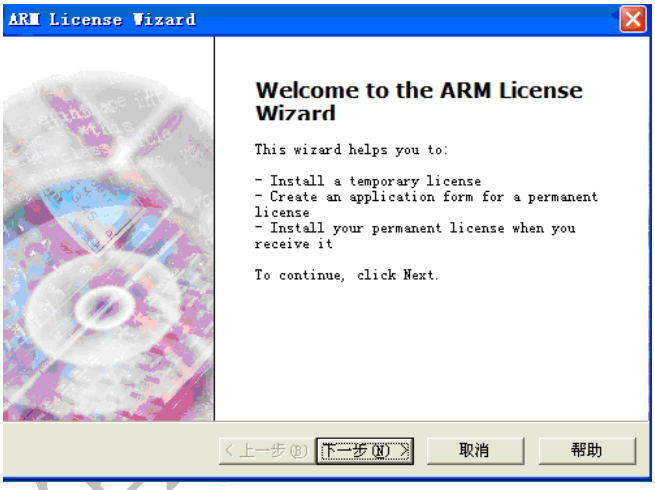

图 2-4　按安装向导安装许可文件

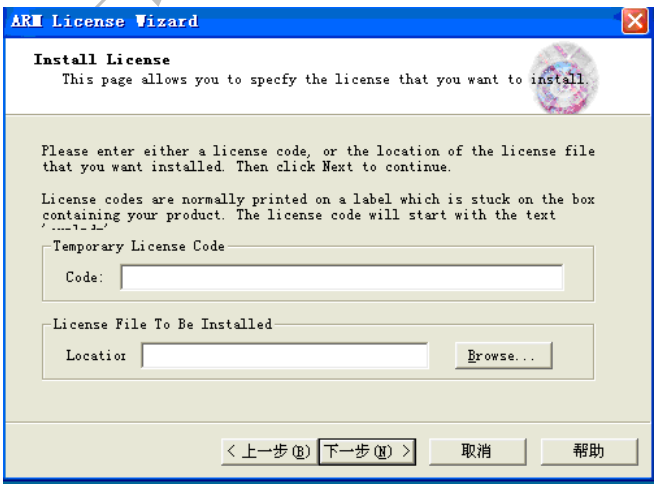

图 2-5　浏览选许可文件

在图 2-5 中选浏览（Browser）查许可文件，在 Program Files\ARM\ADSV1_2\license\中选 license.dat 文件并打开，单击“下一步”按钮，如图 2-6 所示，即可完成 ADS 1.2 的安装。

最后，程序还要注册，注册文件在 Program Files\ARM\ADSV1_2 文件夹中，单击注册文件，即完成程序注册。

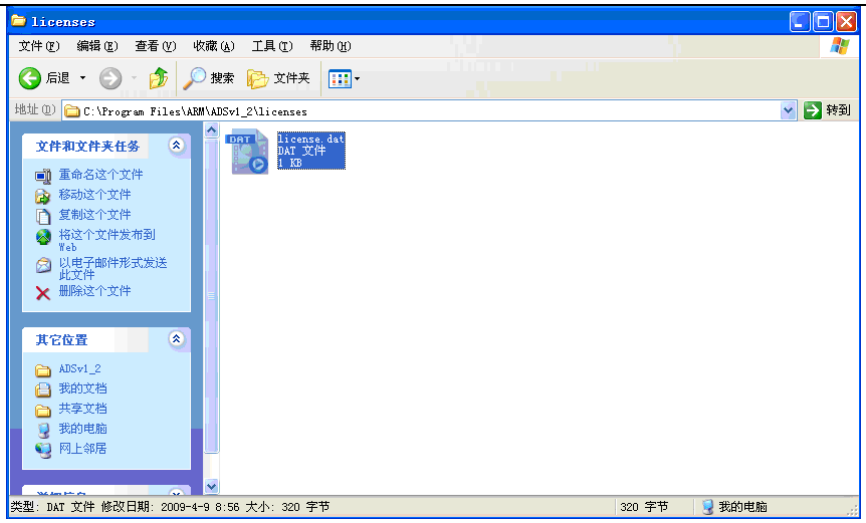

图 2-6　选许可文件

2.2　ADS 集成开发环境的使用

2.2.1　建立一个新工程

运行 ADS 1.2 集成开发环境（CodeWarrior for ARM Developer Suite），单击 File|New 菜单，打开如图 2-7 所示对话框。在 New 对话框 project 选项卡中，共有 7 个选项，选择“ARM Executable Image”ARM 通用模板，即可生成 ARM 的执行文件。

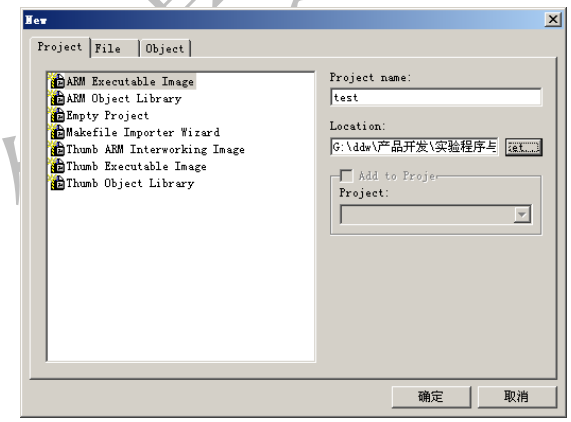

图 2-7　建立一个新工程

还要在“Project name”栏中输入项目的名称，以及在“Location”中输入其存放的位置，单击“确定”按钮保存项目。

2.2.2　开发环境设置

（1）在新建的工程中，选择 Debug 版本，如图 2-8 所示，使用 Edit|Debug Settings 菜单对 Debug 版本进行参数设置。

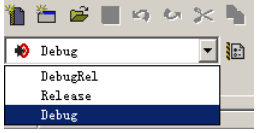

图 2-8　选择 Debug 版本

（2）在如图 2-9 中，单击“Debug Setting”按钮，弹出如图 2-10 所示对话框，选中“Target Setting”选项，在 Post-linker 栏中选中“ARM fromELF”项，单击“OK”按钮确定。这是为生成可执行代码的初始开关。

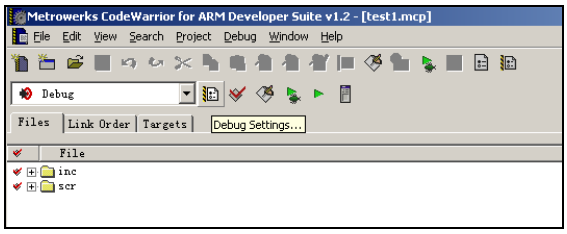

图 2-9　单击“Debug Settings”按钮

（3）在图 2-10 中，单击“ARM Assembler”选项，出现如图 2-11 所示对话框。在“Architecture or Processer”栏中选 ARM920T。这是项目选择的 CPU 类型。

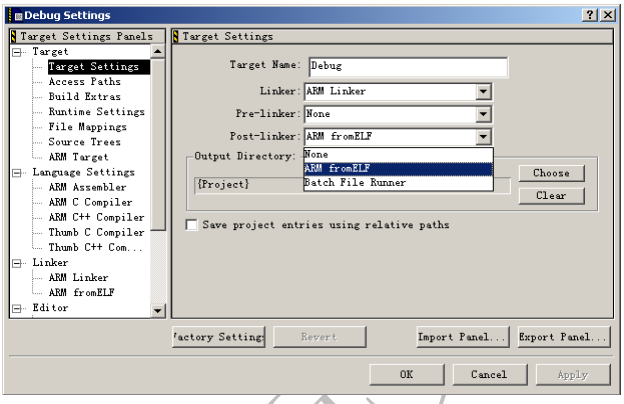

图 2-10　Debug Settings

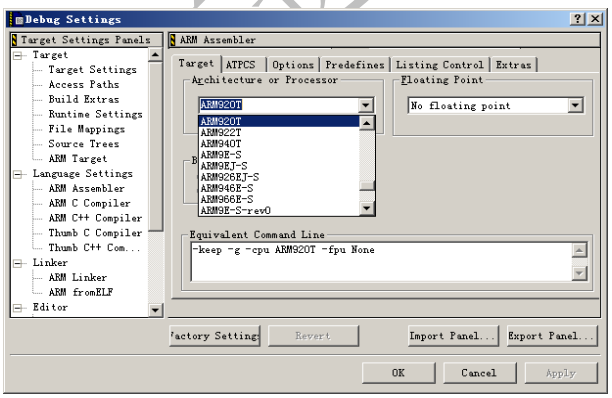

图 2-11　选择 CPU 类型

（4）单击“ARM C Compliler”选项，出现如图 2-12 所示的对话框。在“Architecture or Processer”栏中选 ARM920T，这是要编译的 CPU 核。

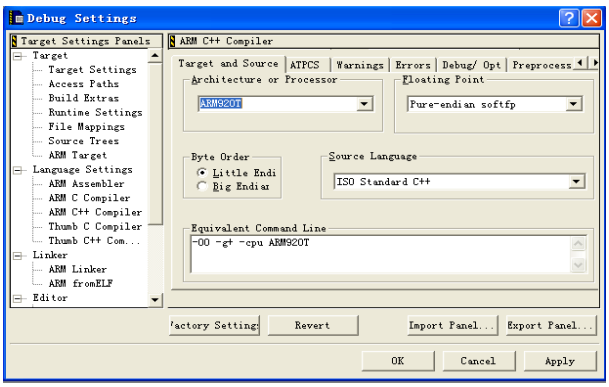

图 2-12　选择要编译的 CPU 核

（5）单击“ARM linker”选项，在“Output”选项卡中设定程序的代码段地址，以及数据使用的地址，如图 2-13 所示。图中的“RO Base”栏中填写程序代码存放的起始地址，“RW Base”栏中填写程序数据存放的起始地址。该地址是属于 SDRAM 的地址。

选择“Options”选项卡，如图 2-14 所示，“Image entry point”栏中要填写程序代码的入口地址，其他保持不变，如果是在 SDRAM 中运行，则可在 0x30000000～0x33ffffff 中选值，这是 64MB SDRAM 的地址，但是这里用的是起始地址，所以必须把你的程序空间给留出来，并且还要留出足够的程序使用的数据空间，而且还必须是 4 字节对齐的地址（ARM 状态）。通常入口点 Image entry point 为 0x30000000，ro_base 也为 0x30000000。

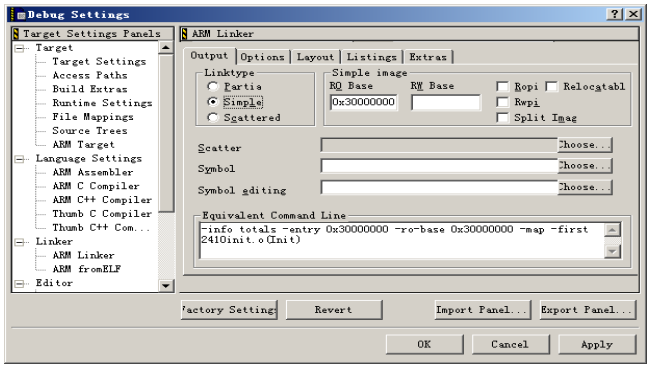

图 2-13　填写程序代码存放的起始地址

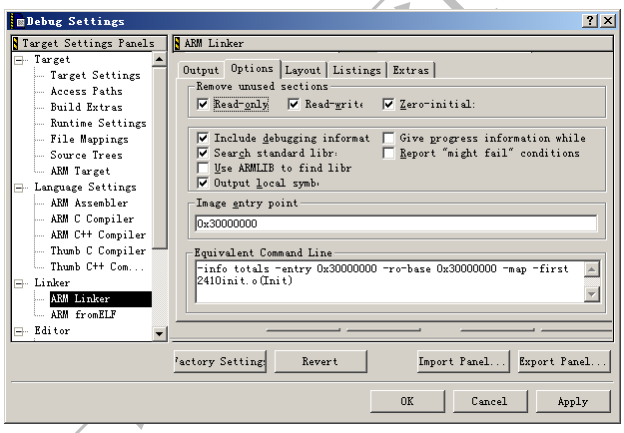

图 2-14　填写程序代码的入口地址

选择“Layout”选项卡，如图 2-15 所示，在“Place at beginning of image”栏内，需要填写项目的入口程序的目标文件名，如整个工程项目的入口程序是 2410init.s，那么应在 Object/Symbol 处填写其目标文件名 2410init.o，在“Section”处填写程序入口的起始段标号。它的作用是通知编译器，整个项目的开始运行是从该段开始的。

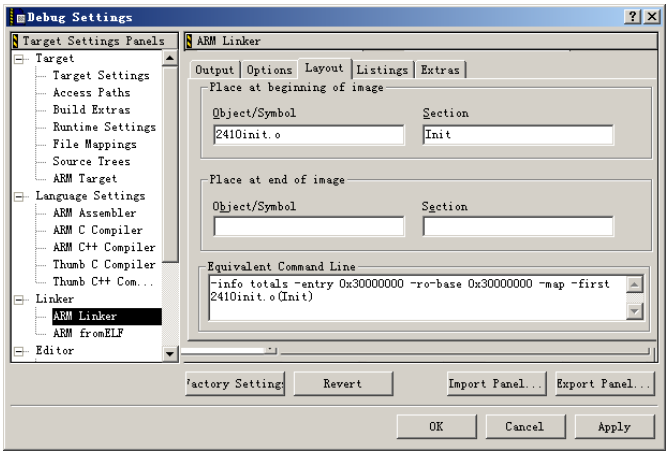

图 2-15　填写项目入口目标文件

（6）在“Debug Setting”对话框中单击左栏的“ARM　fromELF”选项，在“Output file name”栏中设置输出文件名*.bin，前缀名可以自己取，在“Output format”栏中选择“Plain binary”，这是设置要下载到 Flash 中的二进制文件。在图 2-16 中使用的是 test.bin。

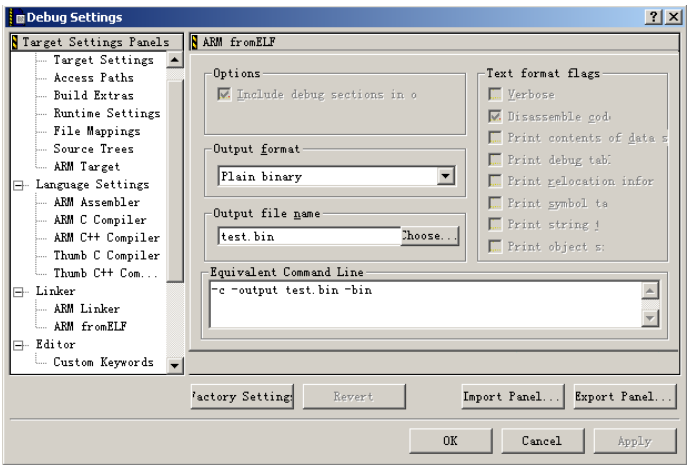

图 2-16　设置输出文件名

（7）到此，在 ADS 1.2 中的基本设置已经完成，按“OK”按钮完成设置。可以将该新建的空的项目文件作为模板保存起来。首先，要将该项目工程文件改一个合适的名字，如 S3C2410 ARM.mcp 等，然后，在 ADS 1.2 软件安装的目录下新建一个合适的模板目录名，如 S3C2410 ARM Executable Image，再将刚刚设置完的 S3c2410 ARM.mcp 项目文件存放到该目录下即可。

（8）新建项目工程后，就可以执行菜单 Project|Add Files 把和工程相关的所有文件加入，ADS 1.2 不能自动进行文件分类，用户必须通过 Project|Create Group 来创建文件夹，然后把加入的文件选中，移入文件夹。或者在文件填加区，单击鼠标右键，如图 2-17 所示。

先选“Add Files”选项，加入文件，再选 Create Group 选项，创建文件夹，然后把文件移入文件夹内。读者可根据自己习惯，更改 Edit|Preference 窗口内关于文本编辑的颜色、字体大小，形状，变量、函数的颜色等设置，如图 2-18 所示。

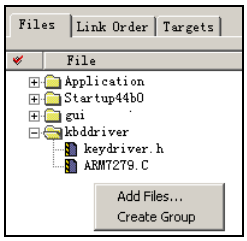

图 2-17　工程所有相关的文件加入

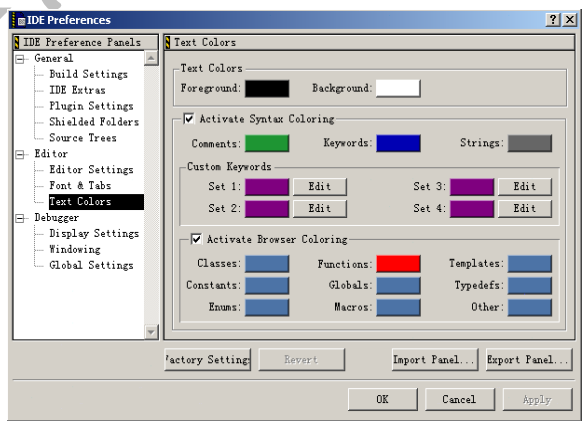

图 2-18　更改文本编辑的颜色

2.2.3　在 ADS 1.2 下进行仿真、调试

在 ADS 1.2 下进行仿真调试，首先需要一根仿真调试电缆和 JTAG 仿真器，用调试电缆把 JTAG 仿真器和上位机并口相连， JTAG 仿真器的驱动程序为两个动态链接库，也要事先安装。

打开调试软件 AXD Debugger。单击 File|Load Image 选项加载可执行文件 xx.axf，打开超级终端，设置其参数为：波特率为 115 200，数据位数为 8，奇偶校验为无，停止位为无 1，数据流控为无。单击“全速运行”按钮，在我们的例子程序中，出现如图 2-19 所示的界面。

最后介绍调试按钮，调试按钮在程序进入 AXD Debugge 状态时会出现在主菜单项中，主要几个调试按钮如图 2-20 所示。

图 2-19　例子程序界面

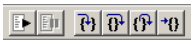

图 2-20　主要几个调试按钮

在图 2-20 中，左起第一个按钮表示全速运行，第二个按钮表示停止运行，第三个按钮表示跳入函数内部单步执行，第四个按钮表示把一个函数作为一个语句单步执行，第五个按钮表示跳出函数。
关于在 ADS1.2 下进行仿真调试，下面还要详细介绍。

2.2.4　其他开发环境介绍

IAR（瑞典爱亚软件技术咨询公司）Embedded Workbench for ARM 是 IAR Systems 公司为 ARM 微处理器开发的一个集成开发环境，下面简称 IAR EWARM。与其他的 ARM 开发环境相比，IAR EWARM 具有入门容易、使用方便和代码紧凑等特点。
IAR Systems 公司目前推出了 IAR Embedded Workbench for ARM version 4.42，并提供一个 32KB 代码限制的学习版或 30 天时间限制的免费评估版，可以到 IAR 公司的网站 www.iar.com/ewarm 下载。
IAR EWARM 中包含一个全软件的模拟程序（simulator）。用户不需要任何硬件支持就可以模拟各种 ARM 内核、外部设备甚至中断的软件运行环境。从中可以了解和评估 IAR EWARM 的功能和使用方法。
IAR Embedded Workbench for ARM version 4.42 是一个针对 ARM 处理器的集成开发环境，包含项目管理器、编辑器、编译连接工具和支持 RTOS（嵌入式实时控制系统）的调试工具，在该环境下可以使用 C/C++ 和汇编语言方便地开发嵌入式应用程序。

2.3　用 AXD 进行代码仿真、调试

2.3.1　AXD 简介

ADX(ARM extended Debugger)是 ADS 软件中独立于 Code Warrior IDE 的图形软件，可从 Code Warrior for ARM Developer Suite 中进入 ADX 进行调试，或在 Windows 下选择“程序”|ARM Developer Suite v1.2|AXD Debugger 命令进入调试。要使用 AXD 必须首先有生成包含调试信息的程序，即由 Code Warrior for ARM Developer Suite 编译生成含有调试信息的可执行 ELF 格式的映像文件（*.axf）。

1．在 AXD 中打开调试文件

在“Code Warrior for ARM Developer Suite”界面中，单击“Debugger”选项进入 AXD 调试界面。
选择 File|Load image 命令，打开“Load image”对话框，找到要装入的.axf 映像文件，单击“打开”按钮，就可以把映像文件装载到目标内存中，如图 2-21 所示。

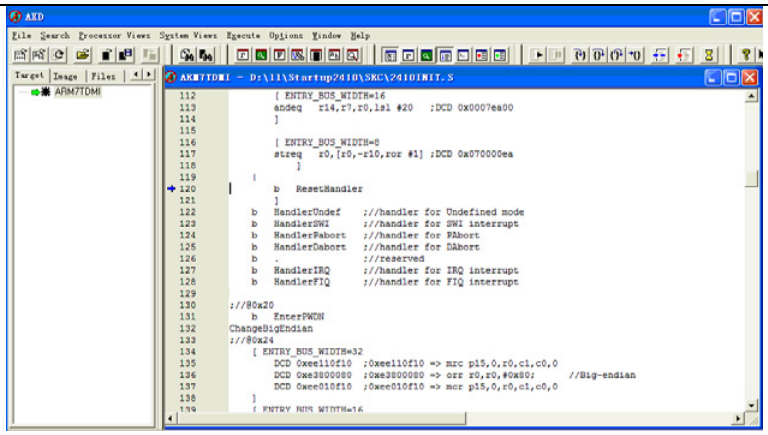

图 2-21　装入 axf 映像文件

利用“Exeute”菜单中的子菜单项对可执行映像文件进行调试，各选项的含义如下。

- 选择“Go”子菜单或按“F5”键，将全速运行代码。
- 选择“Stop”子菜单或按“Shift+F5”键，将停止运行代码。
- 选择“Step In”子菜单或按“F8”键，以单步执行代码，若遇到函数，则进入函数内执行。
- 选择“Step”子菜单或按“F10”键，以单步执行代码，若遇到函数，则把函数看成一条语句单步执行。
- 选择“Step Out”子菜单或按“Shift + F8”键，在“Step In”单步执行代码进入函数内后，若选该子菜单，则可以从函数中跳出返回上一级程序执行。
- 选择“Run To cursor”子菜单或按“F7”键，以全速运行到光标处停下。
- 选择“Show Execution Context”子菜单，可显示执行的内容。
- 选择“Delete All Breakpoint”子菜单，清除所有的断点。

2．查看存储器、寄存器、变量值

利用“AXD 菜单选项“Processor Views”和“System Views”中的子菜单选项可查看寄存器、变量值，还可以查看某个内存单元的数值等，各子菜单的含义如下。

- 选择“Registers”子菜单或按“Ctrl+R”键，可查看或修改目标板处理器中寄存器中的值。
- 选择“Watch”子菜单或按“Ctrl+E”键，可对处理器设置观察点，观察点可以是寄存器、地址等， 但不能修改。特别注意：“Processor Views”菜单下的“Watch”只能观察处理器，而“System Views”菜单下的“Watch”或按“Alt+E”键时可对目标板上的任何资源建立观察，可增加或删除观查点。
- 选择“Variables”菜单或按“Ctrl+E”键，可查看或修改当前可执行的映像文件（程序）中的变量值，这些变量可以是局部变量、全局变量、类属变量。可增加或删除查看或修改的变量。
- 选择“Memory”子菜单或按“Ctrl+M”键，可查看或修改存储器中的值。

如在程序执行前，可以先查看两个宏变量 IOPMOD 和 IOPDATA 的当前值。选择“AXD”的“Processor Views｜Memory”命令或按“Ctrl +M”键后，查看或修改存储器中的值，如图 2-22 所示。

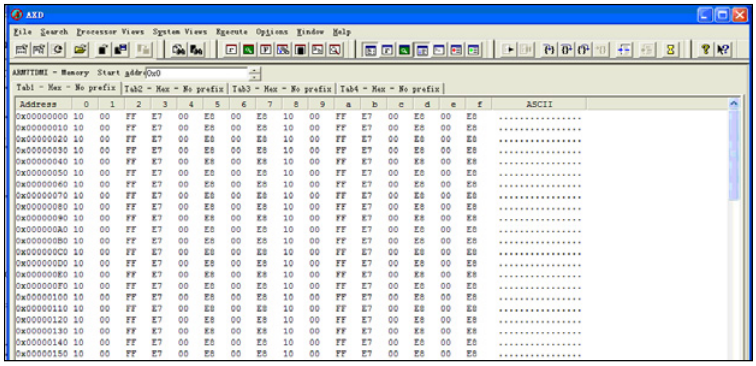

图 2-22　查看或修改存储器中的值

在“Memory Start address”文本框图上，用户可以根据要查看或修改的存储器地址输入起始地址，在下面的表格中会列出连续的 64 个地址。因为 I/O 模式控制寄存器和 I/O 数据控制寄存器都是 32 位的控制寄存器，所以从 0x00000000 开始的连续 4 个地址空间存放的是 I/O 模式控制寄存器的值，从图中可以读出该控制寄存器的值，数据控制寄存器的内容，注意因为用的是小端模式，所以读数据时注意高地址中存放的高字节，低地址存放的是低字节。

3．断点设置、查看

在程序调试时经常设置断点，即在程序的某处设置断点，当程序执行到断点处即可停下，这时开发人员可通过前面的方法查看寄存器、存储器或变量的值，以判定程序是否正常。设置断点的方法是将光标移到需设置断点处，使用快捷键“F9”在此处设置断点。

查看断点的方法是：选择“System Views｜breakpoint view”命令或按“Alt+K”键，在断点状态对话框中单击右键，利用快捷菜单可增加或删除断点。按“F5”键，程序将运行到断点，如果要进入函数内查看，可以选择“Execte｜Step Inw”命令或按下“F8”键，进入到子函数内部进行单步程序的调试。

2.3.2 JTAG 概述

由于 IEEE 1149.1 标准是由 JTAG（Joint Test Action Group，联合测试行动小组）这个组织最初提出的，最终由 IEEE 批准并且标准化的，所以 IEEE 1149.1 这个标准一般称为 JTAG 调试标准。

JTAG 标准主要用于芯片内部测试及对系统进行仿真、调试。JTAG 技术是一种嵌入式调试技术，它在芯片内部封装了专门的测试电路 TAP（Test Access Port 测试访问口），通过专用的 JTAG 测试工具对内部节点进行测试。目前大多数比较复杂的器件都支持 JTAG 协议，如 ARM、DSP、FPGA 器件等。标准的 JTAG 接口是 4 线：TMS、TCK、TDI、TDO，分别为测试模式选择、测试时钟、测试数据输入和测试数据输出。JTAG 测试允许多个器件通过 JTAG 接口串联在一起，形成一个 JTAG 链，能实现对多个器件分别测试。JTAG 接口还常用于实现 ISP（In-System Programmable 在线系统可编程）功能，如对 Flash 器件进行编程等。

在 JTAG 调试中，边界扫描（Boundary-Scan）是一个很重要的概念。边界扫描技术的基本思想是在靠近芯片的输入/输出管脚上增加一个移位寄存器单元。因为这些移位寄存器单元都分布在芯片的边界上，所以被称为边界扫描寄存器（Boundary-Scan Register Cell）。

芯片处于调试状态的时候，这些边界扫描寄存器可以将芯片和外围的输入/输出隔离开来。通过这些边界扫描寄存器单元，可以实现对芯片输入/输出信号的观察和控制。如果需要捕获芯片某个管脚上的输出，首先需要把该管脚上的输出装载到边界扫描链的寄存器单元中去，然后通过 TDO 输出，这样，我们就可以从 TDO 上得到相应管脚上的输出信号。如果要在芯片的某个管脚上加载一个特定的信号，则首先需要通过 TDI 把期望的信号移位到与相应管脚相连的边界扫描链的寄存器单元里去，然后将该寄存器单元的值加载到相应的芯片管脚。

由于在正常的运行状态下，这些边界扫描寄存器对芯片来说是透明的，所以正常的运行不会受到任何影响。这样，边界扫描寄存器就提供了一个便捷的方式，用以观测和控制所需要调试的芯片。另外，芯片输入/输出管脚上的边界扫描（移位）寄存器单元可以相互连接起来，在芯片的周围形成一个边界扫描链（Boundary-Scan Chain）。一般的芯片都会提供几条独立的边界扫描链，用来实现完整的测试功能。边界扫描链可以串行地输入和输出，通过相应的时钟信号和控制信号，可以方便地观察和控制处在调试状态下的芯片。

JTAG 仿真器需要设备驱动程序驱动，在我们使用的教学实验系统（EDUKIT-Ⅲ），JTAG 仿真器的驱动程序为两个动态链接库（EasyICEArm9Plus.dll，EasyICEArm7Plus.dll），把这两个文件复制到 C:\EmbestIDE\Bin\Device\路径下，即可正常使用。

2.3.3 Nor Flash 和 Nand Flash 的区别和使用

Nor 和 Nand 是现在市场上两种主要的非易失闪存技术。Intel 公司于 1988 年首先开发出 Nor Flash 技术。这项技术的开发和投放市场彻底改变了原先由 EPROM 和 EEPROM 一统天下的局面。紧接着，1989 年东芝公司发布了 Nand Flash 结构，强调降低每比特的成本，提供更高的性能，并且像磁盘一样可以通过接口轻松升级。在具有 Nand Flash 接口的系统中，Nand Flash 存储器可以替代 Nor Flash 存储器使用。许多业内人士也搞不清楚 Nand 技术相对于 Nor 技术的优越之处，因为大多数情况下闪存只是用来存储少量的代码，这时 Nor 闪存更适合一些。而 Nand 则是高数据存储密度的理想解决方案。

Nor 的特点是 XIP（eXecute In Place，芯片内执行）特性，这样，应用程序可以直接在 Flash 闪存内运行，不必再把代码读到系统 RAM 中。Nor 的传输效率很高，在 1～4MB 的小容量时具有很高的成本效益，但是很低的写入和擦除速度大大影响了它的性能。

Nand 结构能提供极高的单元密度，可以达到高存储密度，并且写入和擦除的速度也很快。应用 Nand 的困难在于 Flash 的管理和需要特殊的系统接口。

1．性能比较

Flash 闪存是非易失存储器，可以对称作块的存储器单元块进行擦写和再编程。由于任何 Flash 器件的写入操作只能在空或已擦除的单元内进行，所以大多数情况下，在进行写入操作之前必须先执行擦除。Nand 器件执行擦除操作是十分简单的，而 Nor 器件则要求在进行写入前先要将目标块内所有的位都写为 0。

由于擦除 Nor 器件时是以 64KB～128KB 的块进行的，执行一个写入/擦除操作的时间为 5s，与此相反，擦除 Nand 器件是以 8KB～32KB 的块进行的，执行相同的操作最多只需要 4ms。执行擦除时块尺寸的不同进一步拉大了 Nor 和 Nand 之间的性能差距，统计表明，对于给定的一套写入操作，尤其是更新小文件时，在基于 Nor 的单元中进行需要更多的擦除操作。这样，当选择存储解决方案时，设计师必须权衡以下的各项因素。

- Nor 的读速度比 Nand 稍快一些。
- Nand 的写入速度比 Nor 快很多，Nand 的 4ms 擦除速度远比 Nor 的 5s 快。
- 大多数写入操作需要先进行擦除操作。
- Nand 的擦除单元更小，相应的擦除电路更少。

2．容量和成本

Nand Flash 的单元尺寸是 Nor 器件的一半，由于生产过程更为简单，Nand 结构可以在给定的模具尺寸内提供更高的容量，也就相应地降低了价格。在 Nand Flash 中每个块的最大擦写次数是一百万次，而 Nor Flash 的擦写次数是十万次。

Nor Flash 占据了容量为 1MB～16MB 闪存市场的大部分，而 Nand Flash 只是用在 8MB～128MB 的产品当中，这也说明 Nor 主要应用在代码存储介质中，Nand 适合于数据存储。Nand 在 Compact Flash、Secure Digital、PC Cards 和 MMC 存储卡市场上所占份额最大。

3．接口差别

Nor Flash 带有 SRAM 接口，有足够的地址引脚来寻址，可以很容易地存取其内部的每一个字节。基于 Nor 的闪存使用非常方便，可以像其他存储器那样连接，并可以在上面直接运行代码。

Nand 器件使用复杂的 I/O 口来串行存取数据，各个产品或厂商的方法可能各不相同。8 个引脚用来传送控

制、地址和数据信息。Nand 的读写操作采用 512 字节的块，这一点与硬盘管理操作类似，显然基于 Nand 的存储器就可以取代硬盘或其他块设备。

在使用 Nand 器件时，必须先写入驱动程序，才能继续执行其他操作。向 Nand 器件写入信息需要相当的技巧，因为设计师不能向坏块写入，这就意味着在 Nand 器件上自始至终都必须进行虚拟映射。

幸运的是，S3C2410 微处理器支持 Nand Flash 接口，大大方便了在嵌入式系统设计中的应用。鉴于两种存储器各自的优缺点，在 S3C2410 嵌入式系统中，对 Nor Flash 和 Nand Flash 电路都进行了设计，以方便使用。

2.3.4 烧写 Flash

程序调试结束，要将其可执行文件烧写（或称固化）到目标机 Flash 中运行，这个过程要通过一个专门的下载软件来进行，下面以 Embest OnLine Flash Programmer for ARM 为例，来说明该软件的安装和使用。

1．安装 Flash Programmer

Flash Programmer 安装过程比较简单，运行 Flash Programmer 安装包中的 Setup.exe，按照提示一步步执行即可。

Flash Programmer 安装程序将自动区分电脑是否已安装 Embest IDE 软件的情况。

（1）电脑已安装 Embest IDE 软件，安装程序将会把 Flash Programmer 缺省安装到“Embest IDE 安装目录\Tools\FlashProgrammer”目录，如图 2-23 所示。同时安装程序将自动探测是否安装与 IDE 软件共享的设备模块和驱动程序，安装完毕后电脑无需重新启动。如果 IDE 已注册，软件可直接运行。

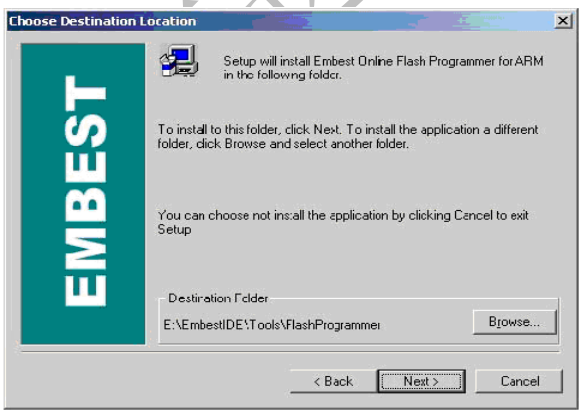

图 2-23　安装 Flash Programmer

（2）电脑未安装 Embest IDE 软件，安装程序将会把 Flash Programmer 缺省安装到“Program Files\Embest\FlashProgrammer”目录，安装完毕后需要重新启动。软件正常运行时需要注册。软件安装完成后将缺省建立“Embest Tools”程序文件夹，包含执行程序和帮助的快捷方式。

2．Flash Programmer 的功能

单击 Flash Programmer 图标，出现图 2-24 所示的对话框，在第一行有 4 个一级菜单，下面分别介绍。

（1）文件菜单。

文件菜单用于保存、打开用户设置的编程配置数据文件，该文件一般以*.cfg 形式存在。通过文件菜单，用户可以将已打开的编程配置数据文件另存为其他文件，还可以打开最近打开过的四个编程配置文件。文件菜单各子菜单命令如表 2-1 所示。

表 2-1　　文件菜单

菜 单 命 令	描　　述
Open	打开编程配置数据文件 *.cfg
Save	保存编程配置数据文件
Save As	另存编程配置数据文件

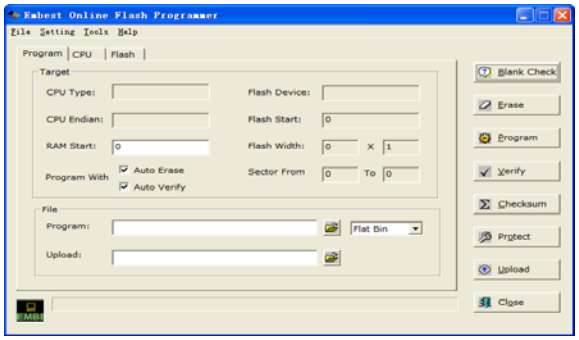

图 2-24 “Flash Programmer”对话框

（2）设置菜单。

设置菜单仅包含“Configure”子菜单，功能如表 2-2 所示。

表 2-2　　设置菜单

菜 单 命 令	描　　述
Configure	选择并配置编程设备

选择“Setting”|“Configure”子菜单，将弹出编程设备配置对话框，如图 2-25 所示。

连接设备（Remote Device）：该下拉框中显示所有本软件支持的编程设备，用户可以选择其中一种，下拉框下方将显示对应该设备的说明和版本。

通信类型（Communication type）：设置该设备与主机使用的连接方式和连接到的主机通信口。

图 2-25　设备配置对话框

通信口（Parallel Port）：按实际连接设置。

（3）“Tools”菜单。

“Tools”菜单仅包含“Option”子菜单，功能如 2-3 所示。

表 2-3　　Tools 菜单

菜 单 命 令	描　　述
Option	配置应用选项

选择“Tools|Option”子菜单，将弹出应用选项对话框，如图 2-26 所示。

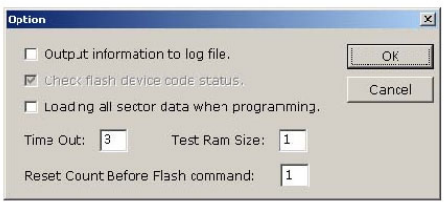

图 2-26　Tools 配置对话框

输出信息记录文件（Output information to log file）：选择该检查框则应用程序将所有输出的提示信息和错误信息记录到安装目录下的 Info.log 文件。

编程时载入扇区所有数据（Loading all sector data when programming）：部分 Flash 芯片编程时需要一次性载入扇区全部数据，则用户可以选择本选项完成编程操作。

超时（Time Out）：设置超时时间，单位为秒。

测试 RAM 区大小（Test Ram Size）：设置目标测试（Target Test）时测试的 RAM 存储区大小，单位为 KB；根据本软件运行时需要，一般情况下应设置为 4KB，对于必须一次性载入一个扇区数据的 Flash 芯片，应该设置为 4KB+扇区大小。

执行 Flash 操作前复位次数（Reset Count Before Flash Command）：设置执行编程、擦除、保护等命令前复位芯片的次数。

（4）“Help”菜单。

“Help”菜单中最重要的是“Contents”子菜单，内部有我们需要的许多资料，特别是用户手册，对我们正确使用 Flash Programmer 有很大帮助、帮助菜单项如表 2-4 所示。

表 2-4　　帮助菜单

菜 单 命 令	描　　述
Contents	显示用户手册
Online Forum	Flash Programmer 在线技术支持论坛
Software Update	访问软件升级通知及本产品最新消息
Register	用户注册
About	关于本软件

3．Flash Programmer 的使用

Net Start 评估板是一款基于 SAMSUNG 公司 ARM7 芯片 S3C4510 的评估电路板，板上包含 2MB 的 Flash 芯片，型号是 AMD 公司 AM29LV160DB，以及 16MB 的 SDRAM，板上应用程序为μcLinux。我们以 Net Start 为例来说明 Flash Programmer 的使用。

Net Start 评估板使用 Flash 前 64KB 存储空间，即 1～4 号扇区保存 BootLoader 软件，该软件用于启动固化在 5～35 号扇区的μcLinux，以及烧写 5～35 号扇区内容。本节讲述如何使用 Flash Programmer 对 Net Start 评估板 Flash 进行编程，用户可以参照本实例对其他电路板进行编程配置。

（1）编程设备设置。

选择“Setting”|“Configure”，弹出编程设备配置对话框，如图 2-27 所示。

选择合适的编程设备并设置通信类型和通信通道。

（2）处理器设置。

选择“CPU”选项卡，如图 2-28 所示。

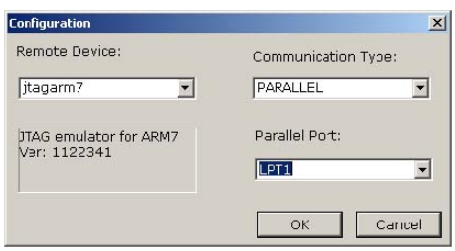

图 2-27　Tools 设备设置对话框

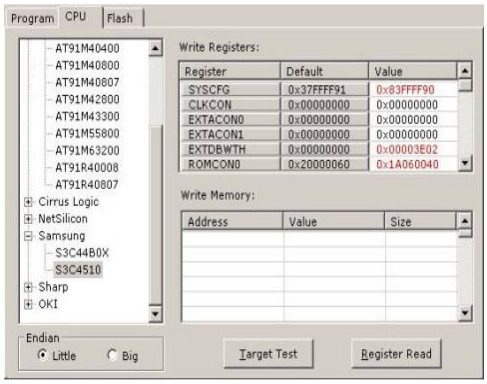

图 2-28　CPU 对话框

选择 SAMSUNG 公司 S3C4510 处理器，选择小端模式，按表 2-5 设置以下寄存器。

表 2-5　　设置以下寄存器

符号	名　称	数值	说　明
SYSCFG	系统配置寄存器	0x03FFFF90	设置内部寄存器基址为 0x3FF0000，基址如果设置为其他地址，必须在写入存储区窗口设置下面的寄存器。选择异步 DRAM 方式
EXTDBWTH	数据总线宽度寄存器	0x00003E02	设置 Flash 0 和 DRAM 0 存储单元访问宽度为 16 位
ROMCON0	Flash 0 号存储单元控制寄存器	0x1A060040	设置 Flash 访问周期，起始地址为 0x180000，结束地址为 0x1A00000
DRAMCON0	DRAM 0 号存储单元控制寄存器	0x10000301	设置为 EDO DRAM，刷新周期为 4 个周期，起始地址为 0x0，结束地址为 0x1000000
REFEXTCON	刷新和扩展 I/O 控制寄存器	0x9C298360	设置刷新参数
INTPND	中断挂起寄存器	0xFFFFFFFF	挂起所有中断，因有复位操作，本寄存器可不设置
INTMSK	中断屏蔽寄存器	0xFFFFFFFF	屏蔽所有中断，因有复位操作，本寄存器可不设置

（3）Flash 芯片设置。

选择“Flash”子对话框，如图 2-29 所示。

选择 AMD 公司 AM29LV160B/DB 芯片，选择访问宽度为 16 位，选择芯片数目为 1 片，设置 Flash 起始地址为为 0x180000，如果要烧写 BootLoader 程序，选择扇区范围为 1～4，如果要烧写μcLinux，选择扇区范围为 5～35。

（4）编程数据设置。

选择“Program”选项卡，如图 2-30 所示。

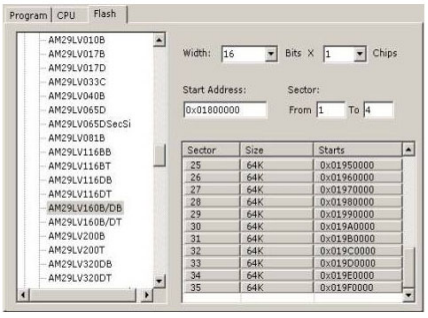

图 2-29　“Flash”对话框

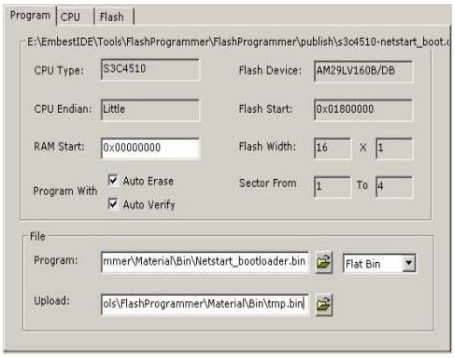

图 2-30　编程数据设置

设置 RAM 起始地址为 0，选择需要编程的 BIN 格式文件，如果需要上载，选择上载文件。

（5）目标板测试。

选择“CPU”选项卡，点击目标板测试“Target Test”按钮，开始目标测试，测试时弹出对话框图 2-31，通知 SYSCFG 寄存器写入值和读取值不相同，原因是 SYSCFG 寄存器中包含 S3C4510 器件标识，因此单击“是”按钮继续测试。弹出类似的对话框均单击“是”按钮继续测试。

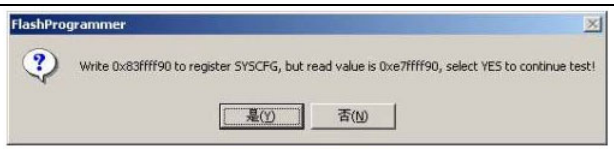

图 2-31　目标板测试对话框

RAM 区测试完后，开始获取 Flash 设备标识号，如果获取的 Flash 标识和保存的标识相同，表示测试成功，输出信息如图 2-32 所示。

图 2-32　表示测试成功对话框

（6）Flash 编程。

用户可以点击编程按钮进行编程操作或其他 Flash 操作。

2.3.5　程序的运行

程序固化到 Flash 中后，运行前往往将其复制到 SDRAM 中去，这样可以提高运行速度，笔者在科研工作中，因工作需要，编写了一段将程序从 Flash 中复制到 DSRAM 中去的 C 语言程序，可供参考，其中 ARM9init(void)程序略。

```
//-------------------------------------------------------------------------
//   主程序
//-------------------------------------------------------------------------
#include "def.h"
#include  "2410addr.h"
#include  "2410lib.h"
#define ARM_ADDR 0X30000000;  // 定义 SDRAM 地址
void (*run)(void );           //定义函数指针
void ARM9init(void);
void CopyFromFlashToRAM(U32 * FlashAddr,U32 *ArmAddr,U32 ul);
void copy(void);
void main(void)
 {
   run=(void(*)(void)) ARM_ADDR;
   ARM9init();
   copy();
   run();
}
//-------------------------------------------------------------------------
//   复制程序
//-------------------------------------------------------------------------
CopyFromFlashToRAM(U32 * pulFlashAddr,U32 *pulArmAddr,U32 ul)
{
  U32 *pulSource=pulFlashAddr;
  U32 *pulDest=pulArmAddr;
  U32 i;
  ul/=4;
  for(i=0;i<ul;i++)
   {
   *pulDest++=*pulSource++;
```

```
  }
}
//-------------------------------------------------------------------------
//   调复制程序
//-------------------------------------------------------------------------
copy(void)
{
     U32*p1;
     U32*p2;
     P1=(U32 * )0x00200000;
     P2=(U32 * )0x30000000;
CopyFromFlashToRAM(p1,p2,0x20000);
}
```

2.4 ARM C 语言程序的基本规则和系统初始化程序

2.4.1 ARM 使用 C 语言编程基本规则

在应用系统的程序设计中，若所有的编程任务均由汇编语言来完成，其工作量巨大，并且不易移植。由于 ARM 的程序执行速度较高，存储器的存储速度和存储量也很高，因此，C 语言的特点充分发挥，使得应用程序的开发时间大为缩短，代码的移植十分方便，程序的重复使用率提高，程序架构清晰易懂，管理较为容易。因此，C 语言的在 ARM 编程中具有重要的地位。

在 ARM 程序的开发中，需要大量读写硬件寄存器，尽量缩短程序的执行时间，因此部分初始化代码使用汇编语言来编写，比如 ARM 的启动代码、ARM 的操作系统的移植代码等，除此之外，大多数代码可以使用 C 语言来完成。

C 语言使用的是标准的 C 语言，ARM 的开发环境实际上就是嵌入了一个 C 语言的集成开发环境，只不过这个开发环境和 ARM 的硬件紧密相关。

在使用 C 语言时，有时要用到和汇编语言的混合编程。当汇编代码较为简洁，则可使用直接内嵌汇编的方法，否则，将汇编程序以文件的形式加入项目当中，通过 ATPCS（ARM/Thumb Procedure Call Standard）的规定与 C 程序相互调用与访问。

ATPCS，就是 ARM、Thumb 的过程调用标准，它规定了一些子程序间调用的基本规则，如寄存器的使用规则、堆栈的使用规则、参数的传递规则等。

在 C 程序和 ARM 的汇编程序之间相互调用必须遵守 ATPCS。而使用 ADS 的 C 语言编译器编译的 C 语言子程序满足用户指定的 ATPCS 的规则。但是，对于汇编语言来说，完全要依赖用户保证各个子程序遵循 ATPCS 的规则。具体来说，汇编语言的子程序应满足下面 3 个条件。

- 在子程序编写时，必须遵守相应的 ATPCS 规则。
- 堆栈的使用要遵守相应的 ATPCS 规则。
- 在汇编编译器中使用-atpcs 选项。

基本的 ATPCS 规定，请参见相关 PDF 文档，简单说明如下。

（1）汇编程序调用 C 程序。

① 汇编程序的设置要遵循 ATPCS 规则，保证程序调用时参数正确传递。

② 在汇编程序中使用 IMPORT 伪指令声明将要调用的 C 程序函数。

③ 在调用 C 程序时，要正确设置入口参数，然后使用 BL 调用。

（2）C 程序调用汇编程序。

① 汇编程序的设置要遵循 ATPCS 规则，保证程序调用时参数正确传递。

② 在汇编程序中使用 EXPORT 伪指令声明本子程序，使其他程序可以调用此子程序。

③ 在 C 语言中使用 extern 关键字声明外部函数（声明要调用的汇编子程序）。

在 C 语言的环境内开发应用程序，一般需要一个汇编的启动程序，从汇编的启动程序跳到 C 语言下的主程序，然后，执行 C 程序，在 C 环境下读写硬件的寄存器，一般是通过宏调用，在每个项目文件的 Startup2410/INC 目录下都有一个 2410addr.h 的头文件，那里面定义了所有关于 2410 的硬件寄存器的宏，对宏读写，就能操作 2410 的硬件，具体的编程规则同标准 C 语言。

2.4.2 初始化程序和开发环境设置

基于 ARM 芯片的应用系统，多数为复杂的片上系统，在系统中，多数硬件模块都是可配置的，需要由软件来预先设置其需要的工作状态，因此在用户的应用程序之前，需要由专门的一段代码来完成对系统基本的初始化工作。由于此类代码直接面对处理器内核和硬件控制器进行编程，故一般均用汇编语言实现。系统的基本初始化内容如下。

- 分配中断向量表。
- 初始化存储器系统。
- 初始化各工作模式的堆栈。
- 初始化有特殊要求的硬件模块。
- 初始化用户程序的执行环境。
- 切换处理器的工作模式。

此外，还要对项目的交叉编译环境进行设置，这其中包括处理器设置、仿真器设置和调试设置等 20 几个大项、近 100 个小项。

系统的初始化程序和交叉编译环境设置是初次学习 ARM 程序设计最难掌握的内容之一，初次学习 ARM 程序设计，有哪些硬件模块需要预先设置其需要的工作状态，如何设置？

初始化程序代码直接面对处理器内核和硬件控制器进行编程，故一般均用汇编语言实现，初学者对 ARM 的汇编语言不熟；交叉编译环境设置大约有 20 几个大项、近 100 个小项要设置，如何保证各项都设置正确，这些都是我们需要解决的问题。

最简单的做法是我们打开一个和我们开发的项目相近的系统提供的例子项目，在例子项目中，系统的初始化程序和交叉编译环境都是设置好的，我们可先不去修改，而是先保留，我们只是把其中的主程序内容换成我们新项目内容，把我们项目所需函数加入进去，这样，就绕过了系统初始化程序和交叉编译环境设置的困惑，节省系统开发时间。

比如，我们要开发一个 LCD 显示项目，项目的初始化程序可能很难，简单作法是我们打开一个例子项目 lcd.mcp，如图 2-33 所示，在项目窗口我们看到，初始化程序在一个文件夹组 startup2410 中，其中包括 3 个汇编语言程序 2410INIT.S、OPTION.S 和 2410SLIB.S，2 个 C 语言程序 2410LIB.C 和 TARGET.C，4 个 C 语言头文件 2410addr.h、2410lib.h、2410slib.h 和 option.h，很复杂。

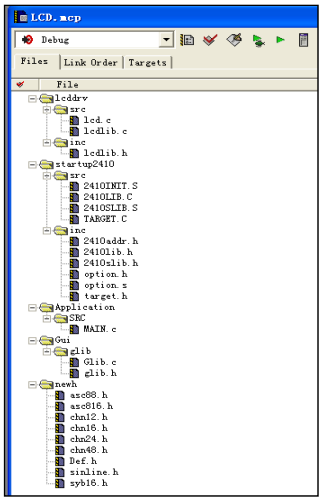

图 2-33 lcd.mcp 的项目窗口

现在我们先不去研究这些程序，把 startup2410 文件夹保留，只修改 Application 文件夹中的 MAIN.C，把我们项目所需要函数加到其中一个文件夹或新建一个文件夹加到项目中，可以很快完成嵌入式系统开发。

2.5 习题与练习

1．在你的计算机中安装 ADS 1.2，并建立一个新项目。
2．如何简化系统初始化程序和开发环境设置，快速完成嵌入式控制系统设计？
3．编写一个 C 语言程序，将用户程序从 Flash 复制到 SDRAM，并从 SDRAM 中运行。
4．Nor Flash 和 Nand Flash 的区别和使用有哪些不同？
5．描述 Flash Programmer 软件的安装和使用。
6．参考图 2-23，分析 lcd.mcp 项目包括几个文件夹，哪个文件夹包含系统初始化程序？
7．系统初始化应完成哪些工作？

联系方式