立宇泰电子 Liyutai Elec. CO., Ltd. 专注于做最好的嵌入式计算机系统供应商

一步一步基于ADS1.2 进行开发（ARM9）

使用说明

Rev 1.0
2007 年 4 月 10 日

杭州立宇泰电子有限公司
HangZhou LiYuTai Elec.Co.,Ltd

目　录

1. ARM 开发环境介绍

1.1 ADS1.2 集成开发环境简介与安装

ADS1.2 是一个使用方便的集成开发环境，全称是 ARM Developer Suite v1.2 。它是由 ARM 公司提供的专门用于 ARM 相关应用开发和调试的综合性软件。在功能和易用性上比较 SDT 都有提高，是一款功能强大又易于使用的开发工具。以下就我们对 ADS1.2 进行一些简要的介绍。

ADS 囊括了一系列的应用，并有相关的文档和实例的支持。使用者可以用它来编写和调试各种基于 ARM 家族 RISC 处理器的应用。你可以用 ADS 来开发、编译、调试采用包括 C、C++和 ARM 汇编语言编写的程序。

ADS 主要由以下部件构成：

◇ 命令行开发工具；

◇ 图形界面开发工具；

◇ 各种辅助工具；

◇ 支持软件。

其中重点介绍一下图形界面开发工具。

◇ AXD 提供给基于 Windows 和 UNIX 使用的 ARM 调试器。它提供了一个完全的 Windows 和 UNIX 环境来调试你的 C、C++和汇编语言级的代码。

◇ Code Warrior IDE 提供基于 Windows 使用的工程管理工具。它的使用使源码文件的管理和编译工程变得非常方便。但 CodeWarrior IDE 在 UNIX 下不能使用。

运行开发板配套光盘中的开发工具下的 ADS1.2 文件夹下面的可执行安装程序 setup.exe，然后就像安装其他 Windows 应用程序一样一步一步向下执行，最后装好 license.dat 文件，编译调试环境就安装好了。

1.2 JTAG 调试代理软件的安装与使用

1.2.1 H-JTAGg 安装。

首先从 H-JTAG 官方网站 http://www.hjtag.com/download.html 中下载最新版本的 H-JTAG。解压后双击安装文件 H-JTAG V0.4.4.EXE（0.4.4 版本的可能不是最新的）双击后进入图 1 安装画面

图 1-1 H-JTAG 安装界面

选择好路径后就一直安装到 finish 安装就 over 了。同时你的桌面会多出两个 和 图标，双击图标就可以运行 H-JTAG 了。

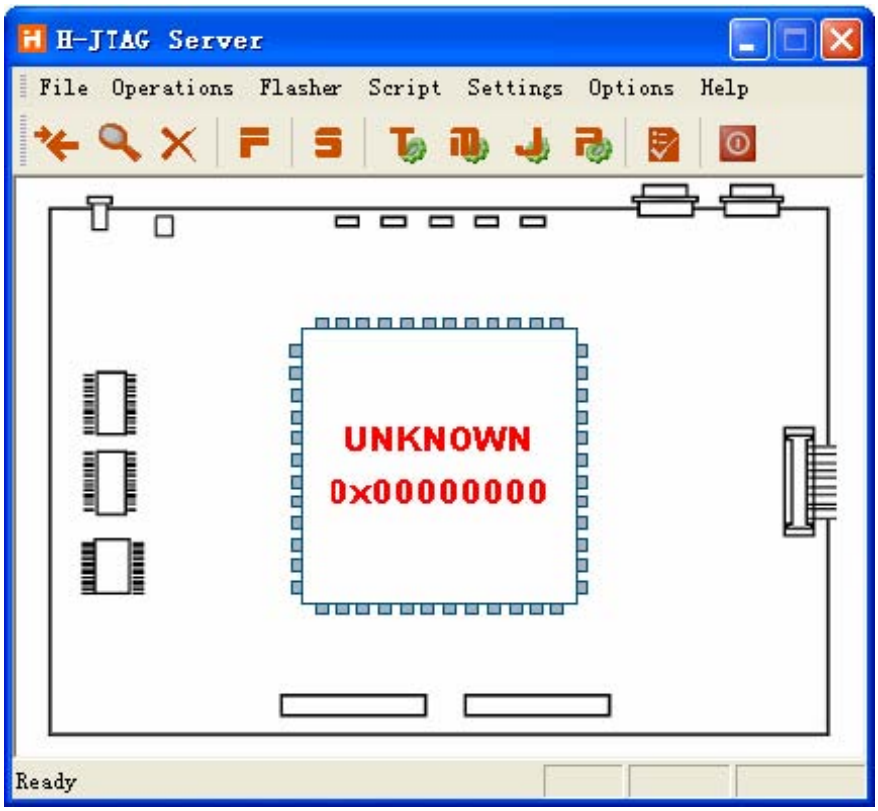

图 1-2 H-JTAG 没有检测到 ARM 内核界面

1.2.2 H-JTAG 设置

首先设置并口，在 Setting>port Setting(当然点 图标效率会高一点)显示如下

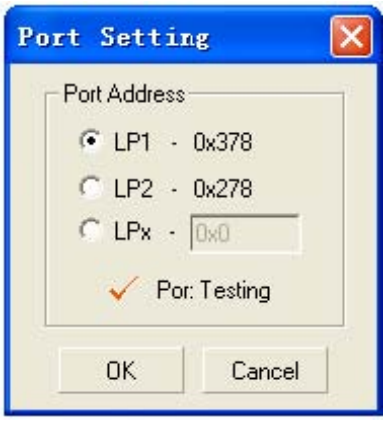

图 1-3 并口设置

根据你计算机的并口具体情况来设置。

接下来设置的是 JTAG 引脚配置(这里很重要，设置不好就有可能检测不到内核)具体先选择 Setting>Jtag Setting（也可以点击 ）出现以下窗口：

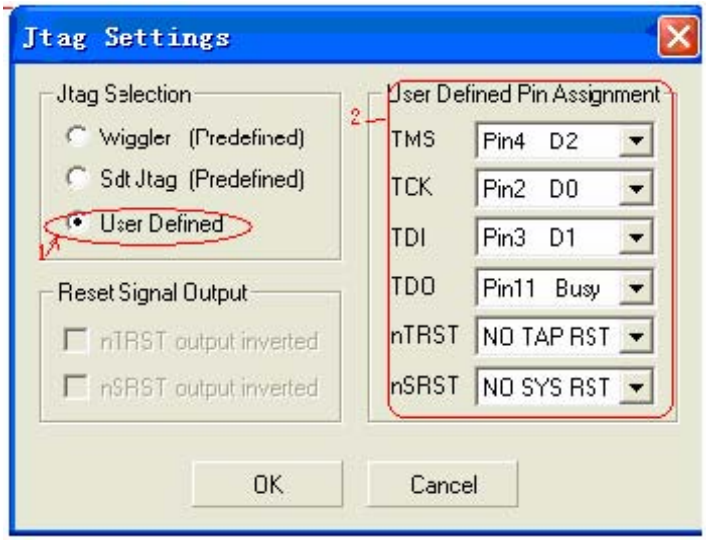

图 1-4H-JTAG 设置

在 Jtag Selection 选择 User Defined(标号 1)，然后在右边设置 Jtag 的引脚（如图中标号 2）

1.TMS=Pin4 D2;

2.TCK=Pin2 D0;

3.TDI=Pin3 D1;

4.TDO=Pin11 Busy;

5.nTRST=NO TAP RST

6.nSRST=NO SYS RST。

完成了引脚的配置之后还需要配置目标板的一些参数。点击 Setting>Target Setting(也可以点击)出现以下窗口

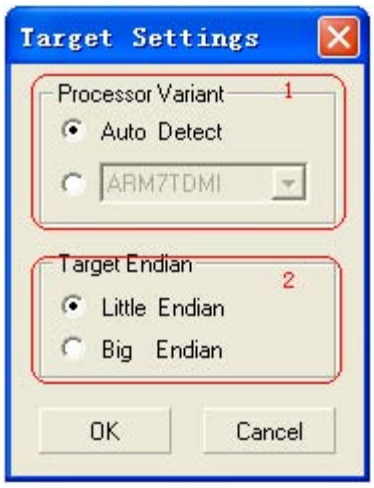

图 1-5 Target 设置

标志 1 是配置处理器变量情况，我们在此选择 Auto Detect（自动检测）；标志 2 是存储格式，我们选择是 Little Endian(小端格式)。

连接好计算机和目标板之后上电，点击 Operations>Detect Target(也可以点击)

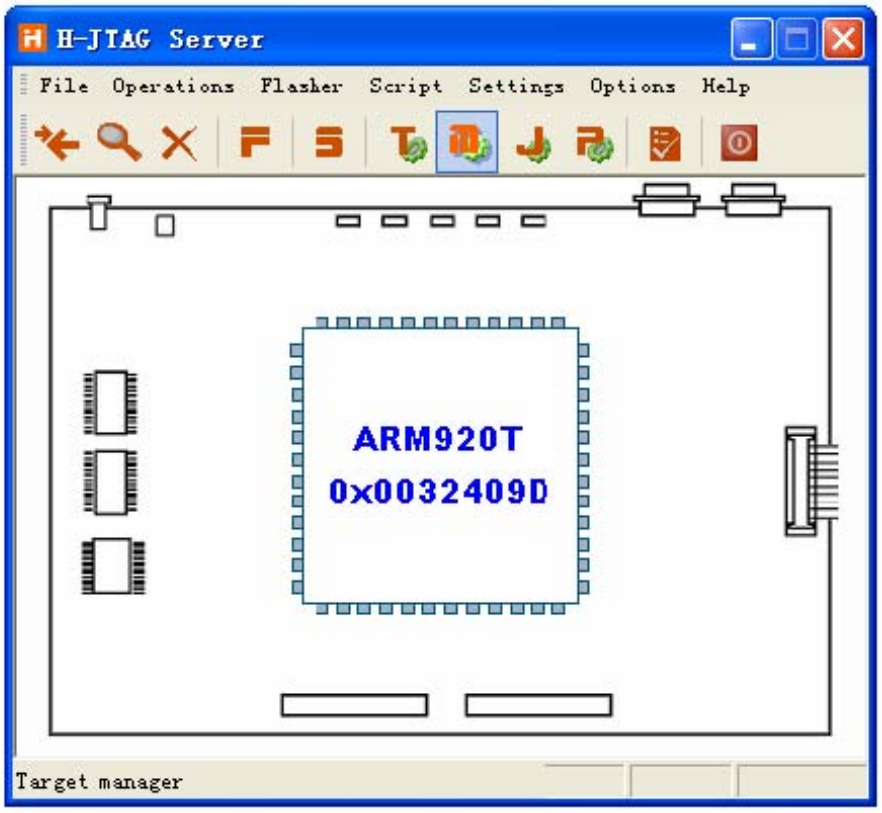

图 1-6 H-JTAG 运行界面

出现上面的画面就说明配置成功（如果没成功的话再看看硬件连接有没有对）。连接完成后，接

下来就可以运行 ADS 了。

2 使用 CodeWarrior 建立工程并进行编译

首先我们学习如何使用 ADS 中的 CodeWarrior ——项目管理器来管理源代码。一个嵌入式系统项目通常是由多个文件构成的，这其中包括用不同的语言（如汇编或 C）、不同的类型（源文件，或库文件）的文件。CodeWarrior 通过“工程（Project）”来管理一个项目相关的所有文件。因此，在我们正确编译这个项目代码以前，首先要建立“工程”，并加入必要的源文件、库文件等。

2.1 建立项目

按照以下步骤来新建一个工程：

(1)选择 File 菜单下的 new 选项，或直接单击  ，出现以下对话框：

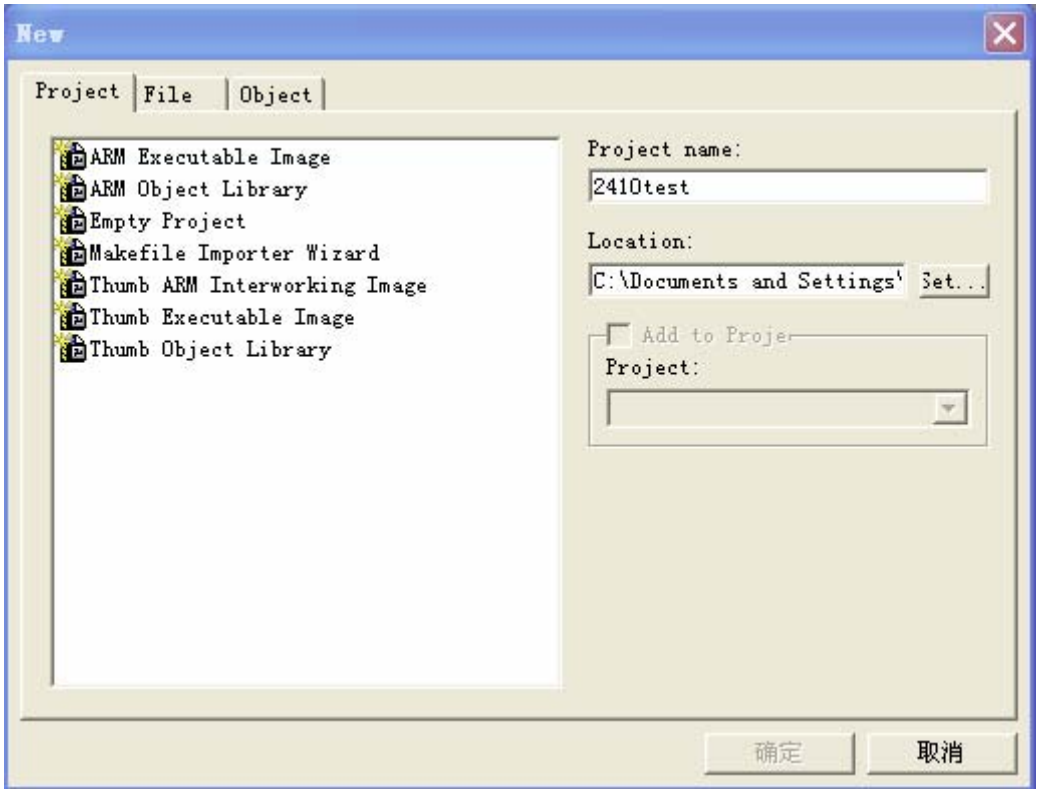

图 2－1 新建工程对话框

（2）选中“ARM Executable Image”选项，在右边的编辑框中输入工程名（例如 2410test），在下面的 Location 栏中，点击“Set…”，选择放置工程的路径。ADS1.20 不支持中文的目录名字，所以新建工程的文件夹向上一直到根目录的所有文件夹的名字都是英文的。

（3）点击[确定]后工程被建立。

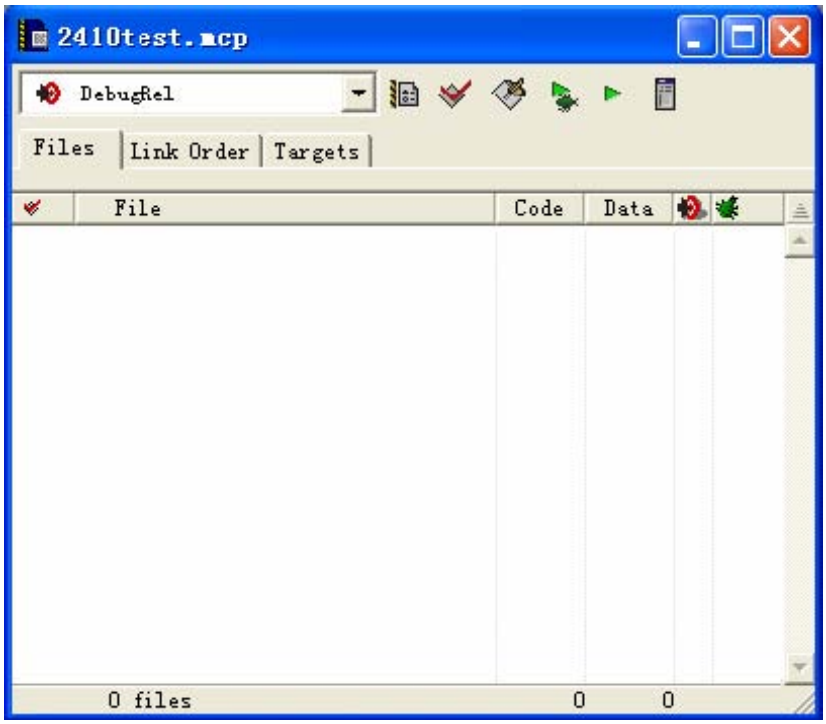

图 2－2 新建工程

但这样的工程还并不能正确地被编译，还需要对工程的编译选项进行适当配置。为了设置方便，先点选 Targets 页面，选中 DebugRel 和 Relese 变量，按下 Del 键将它们删除，仅留下供调试使用的 Debug 变量。点击菜单[Edit | Debug Setting…],弹出配置对话框：

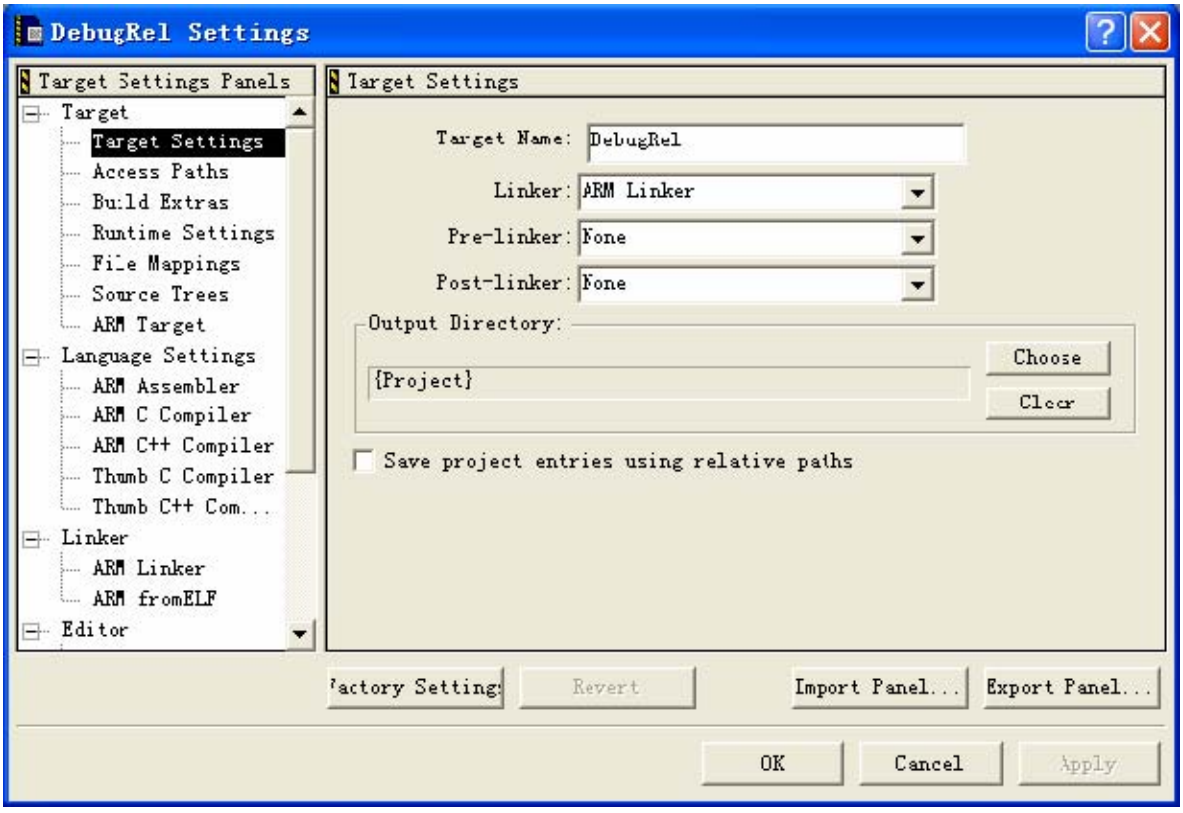

图 2－3 工程配置对话框——目标设置

首先选中 Target Setting ，将其中的 Post-linker 设置为 ARM fromELF ，使得工程在链接后再通过 fromELF 产生二进制代码。

然后选中 ARM Linker，对链接器进行设置：

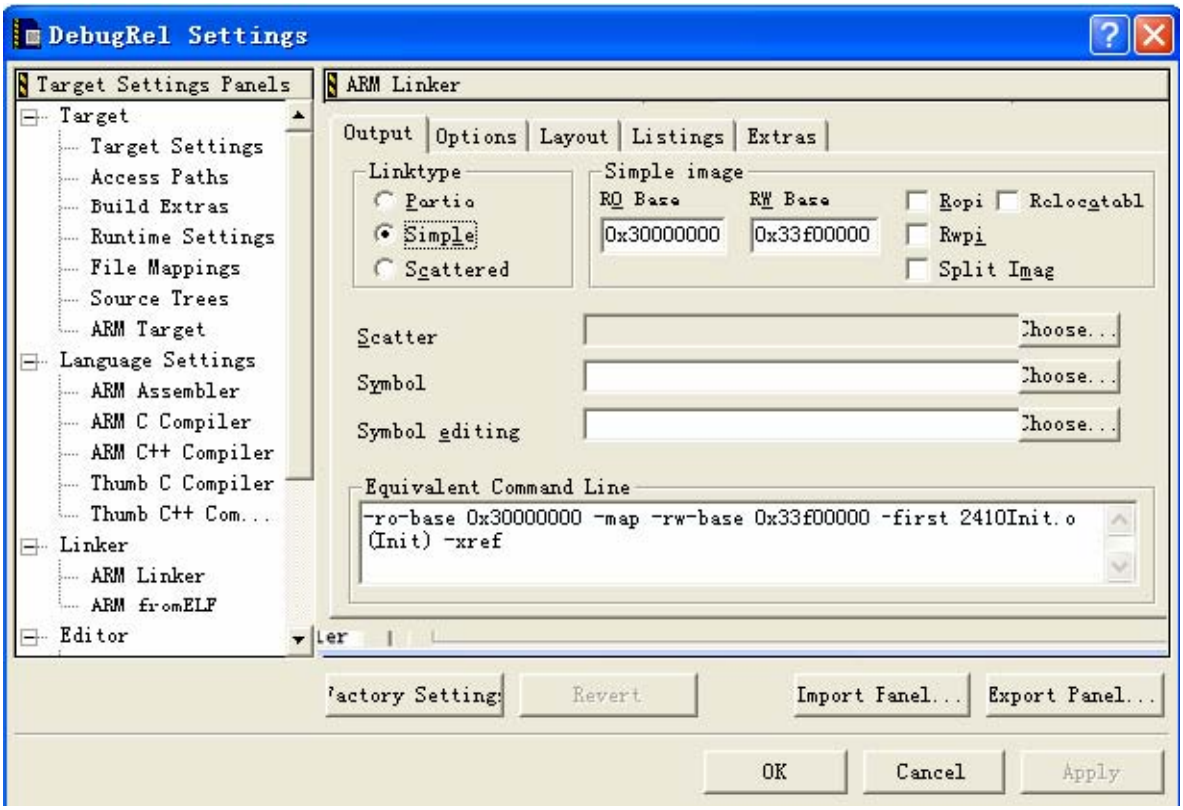

图 2－4（a）ARM Linker 的设置

注意，在调试时，-RO-Base 的设置应当大于 0x30000000 。

选取 Layout 页面进行设置：

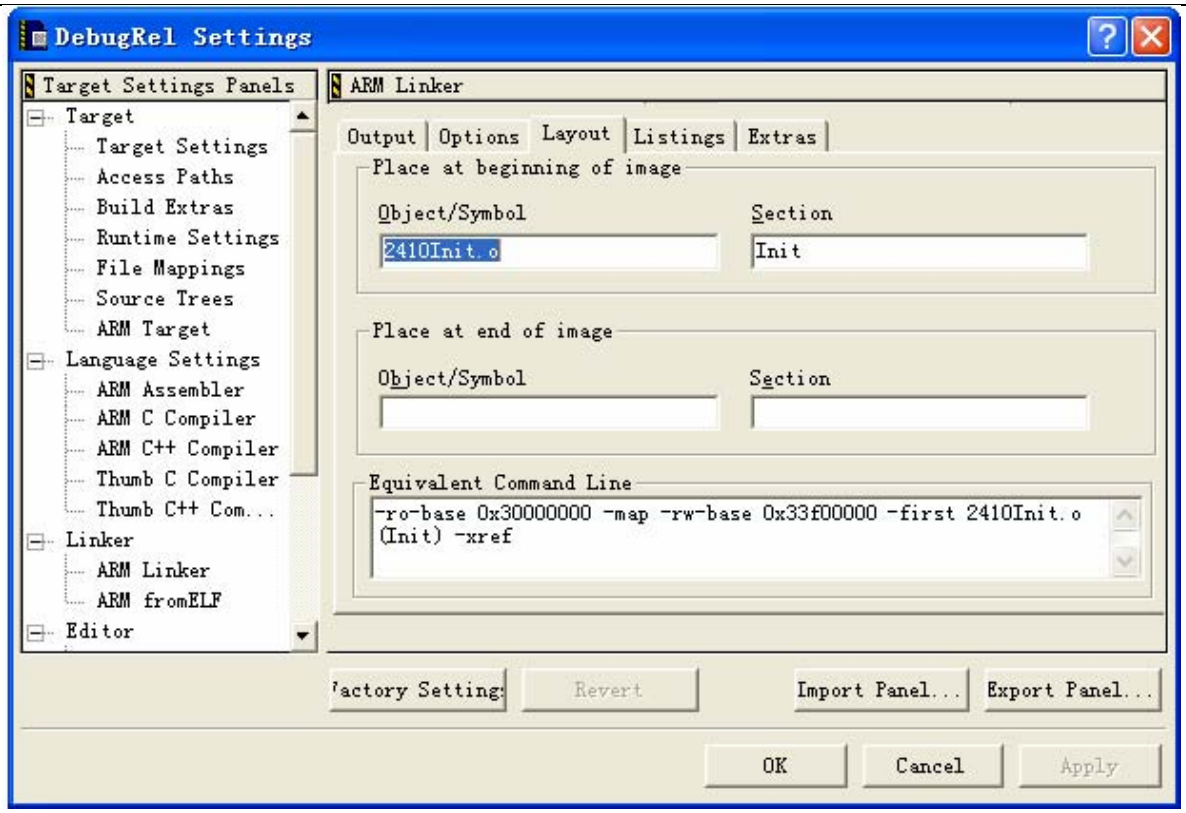

图 2－4（b）ARM Linker 的设置

将 2410init.o 放在映象文件的最前面，它的区域名是 Init。

最后，如果你希望编译的最后生成二进制文件，就要设置 ARM fromELF：

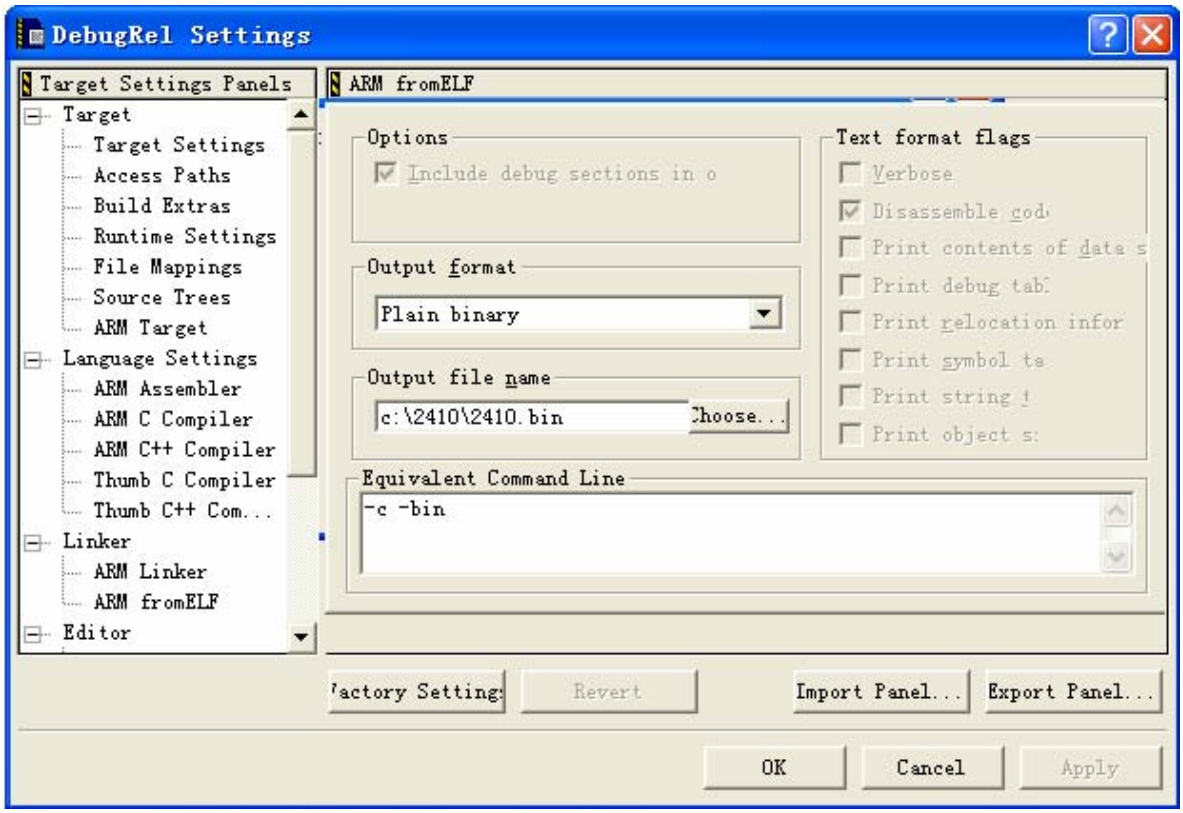

图 2－5 ARM fromELF 的设置

在 Output format 栏中选择 Plain binary ，在 Output file name 栏中，点击“Choose… ”选择你要输出的二进制文件的文件名和路径。

这样，对于 Debug 变量的基本设置都完成了。点击“OK”键退出。

2.2 在工程中添加源文件

在图 4-2 的对话框中，点选 File 页面，选中 Text File，并设置好文件名和路径，点击确定，CodeWarrior 就会为你新建一个源文件，并可以开始编辑该空文件。CodeWarrior 与 SDT 中的 APM 不同，它具有一个很不错的源代码编辑器，因此，大多数时候，我们可以直接采用它的代码编辑器来编写好程序，然后再添加到工程中。

添加源文件的步骤如下：例如添加 2410test.c 文件，在图 4－3 窗口中点选 Files 页面，在空白处单击鼠标右键，点选“Add Files”项，从目录中选取 2410test.c 文件（2410test\2410test.c），点击“打开”，2410test.c 文件就被加入了工程中。

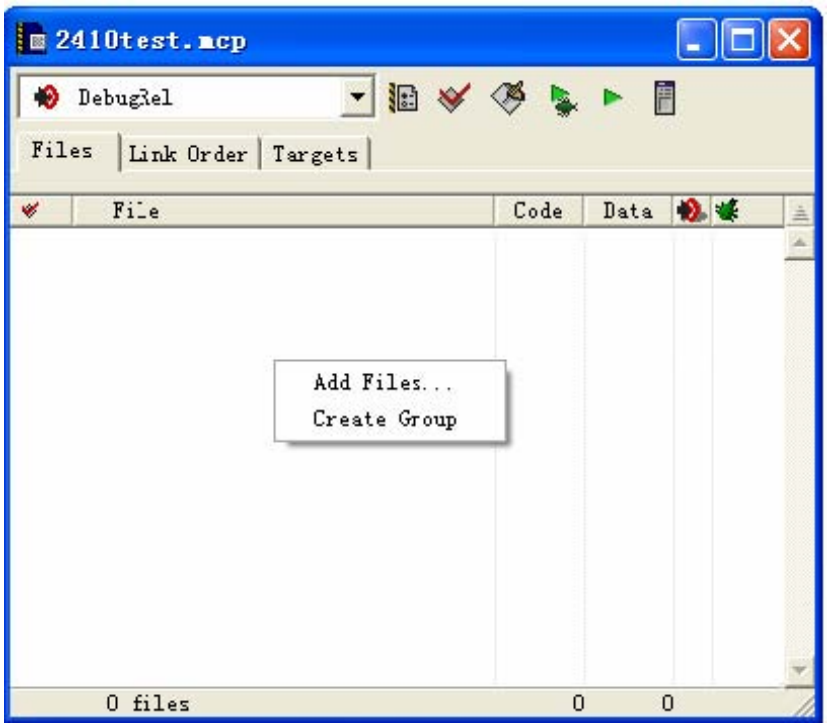

图 2－6 添加源文件

用同样的方法，将 2410test \下所有的*.C 和*.S 源文件文件都添加到 source 中去(包括 Target 目录下的源文件)。

所有必须的文件添加完成后如图 4－8 所示。

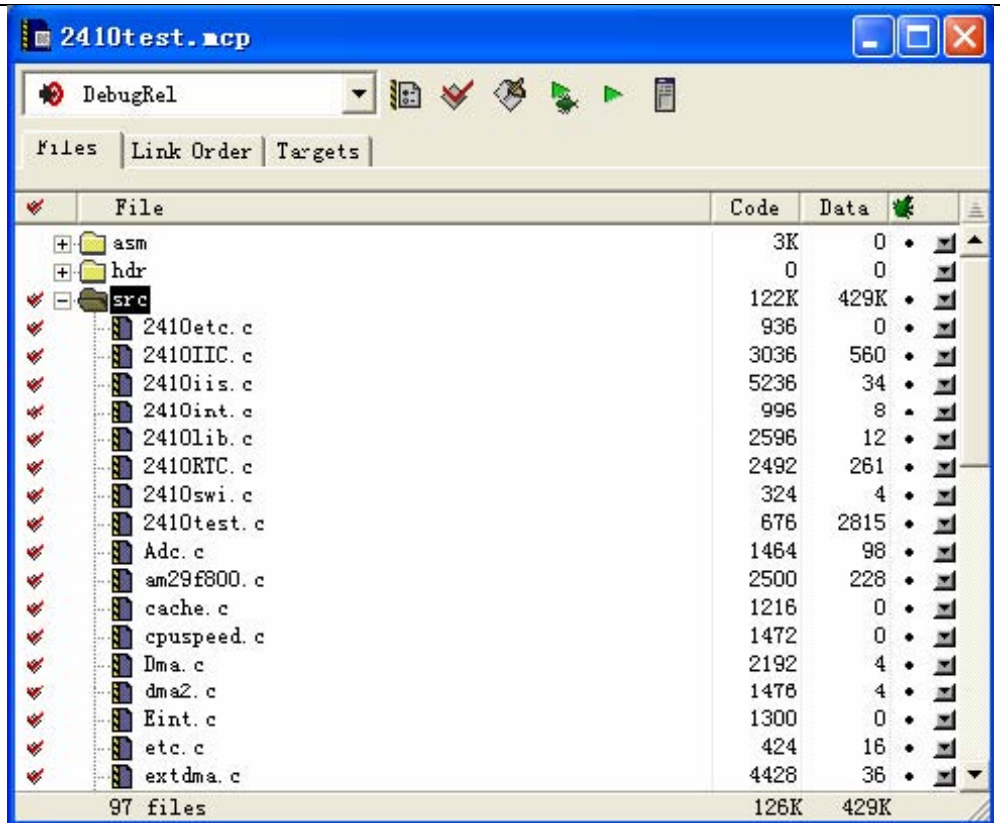

图 2－7 源文件添加完成

2.3 工程进行编译和连接

注意到在上图 4－8 中新加入的文件前面有个红色的“钩”，说明这个文件还没有被编译过。在进行编译之前，你必须正确设置该工程的工具配置选项。如果前面采用的是直接调入工程模板，有些选项已经在模板中保存了下来，可以不再进行设置。如果是新建工程，则必须按照上文中所述的步骤进行设置。

□ 选中所有的文件，点击  图标进行文件数据同步；

□ 然后点击  图标，对文件进行编译（compile）；

□ 点击  按钮，对工程进行 Make，Make 的行为包括以下过程：

○ 编译和汇编源程序文件，产生*.o 对象文件；

○ 链接对象文件和库产生可执行映像文件；

○ 产生二进制代码。

Make 之后将弹出“Errors & Warnings ”对话框，来报告出错和警告情况。编译成功后的显示如下。注意到左上脚标示的错误和警告数目都是“0”，如图所示：

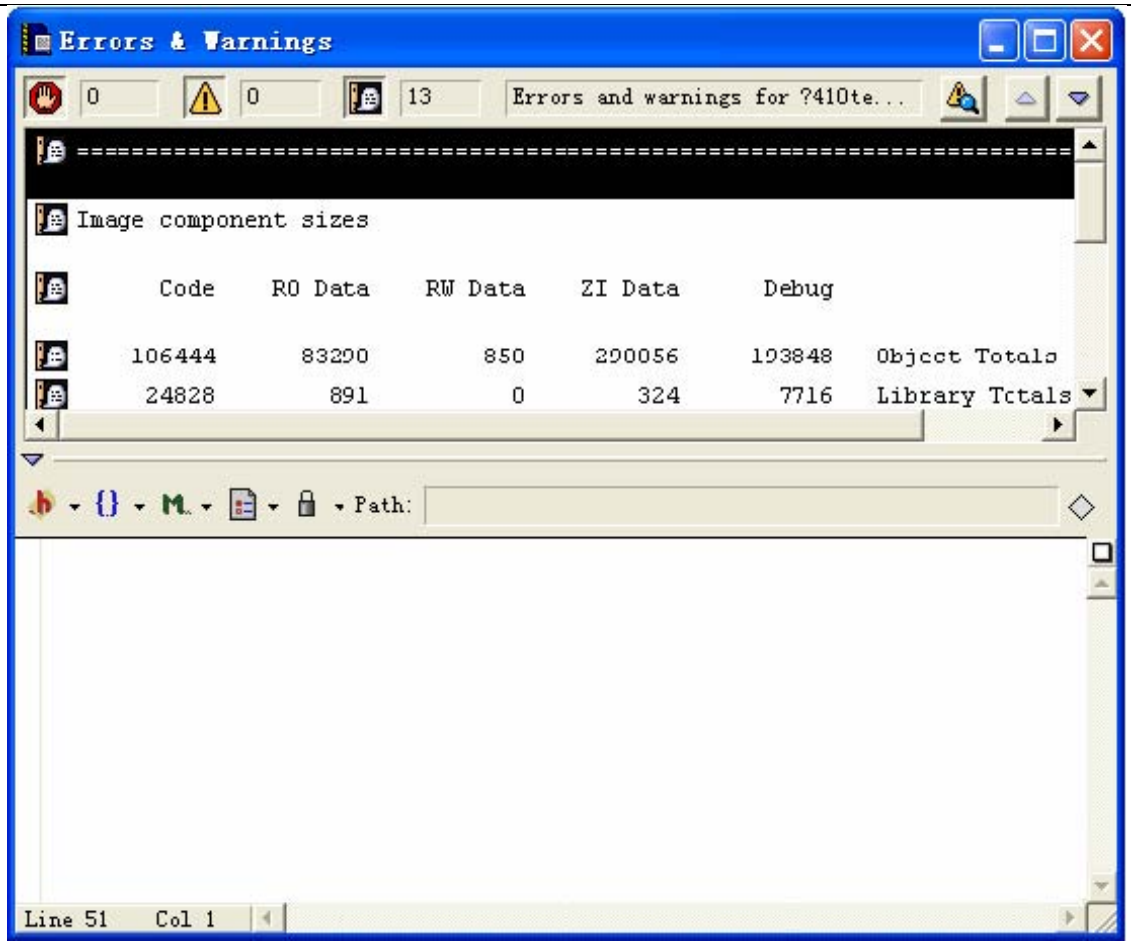

图 2－8 编译后的结果

Make 结束后产生了可执行映象文件 2410test.axf 文件，这个文件可以载入 AXD 进行仿真调试了。并且还通过 fromelf 工具将 ELF 文件转换为二进制格式文件 2410test.bin 。它可以用来最终固化到 flash ROM 中（但链接选项中的-ro-base 要修改为 0x0），也可以下载到-ro-base 地址中运行。

3 使用 AXD 进行仿真调试

3.1 调试前的准备

在调试之前，我们先用并口电缆将计算机并口和 JTAG 仿真小板连接起来，并把 JTAG 仿真小板和开发板的 JTAG 接口用 20 芯排线连接，用串口线将计算机串口和主板的 UART0 口连接起来。然后检查 JTAG 同步复位跳线帽是否插上，所有连线连接正确、接触良好后就可以拨动电源开关上电了。

在打开电源之前，要先打开按上文介绍建立的“超级终端”才能观察到开发板的启动输出信息。电源打开之后，可以听到主板发出一声蜂鸣器的“嘀——”声，看到核心板上绿色发光管 D1、D4 点亮后熄灭，绿色发光管 D2、D3 点亮，D1、D4 和 D2、D3 周期点亮，这说明开发板启动正常。此时点击 H-JTAG 软件 Operations>Detect Target(也可以点击)，如果显示如图 1-6 所示，说明并口已经连接

好了。

在进行调试之前，要先建立好 AXD 与目标系统之间的通讯。如果采用简易 JTAG 调试器进行调试，则首先要运行 H-Jtag。注意，在 AXD 调试器在线仿真期间，不要关闭 H-Jtag。

3.2 AXD 调试器的设置

在 CodeWarrior 编译环境中，工程经过编译成功，产生了*.axf 文件之后，就可以进行调试了。点击  按钮，进入了 AXD 视窗界面。点击菜单项[Option | Configure Target…] ，对调试目标进行配置：

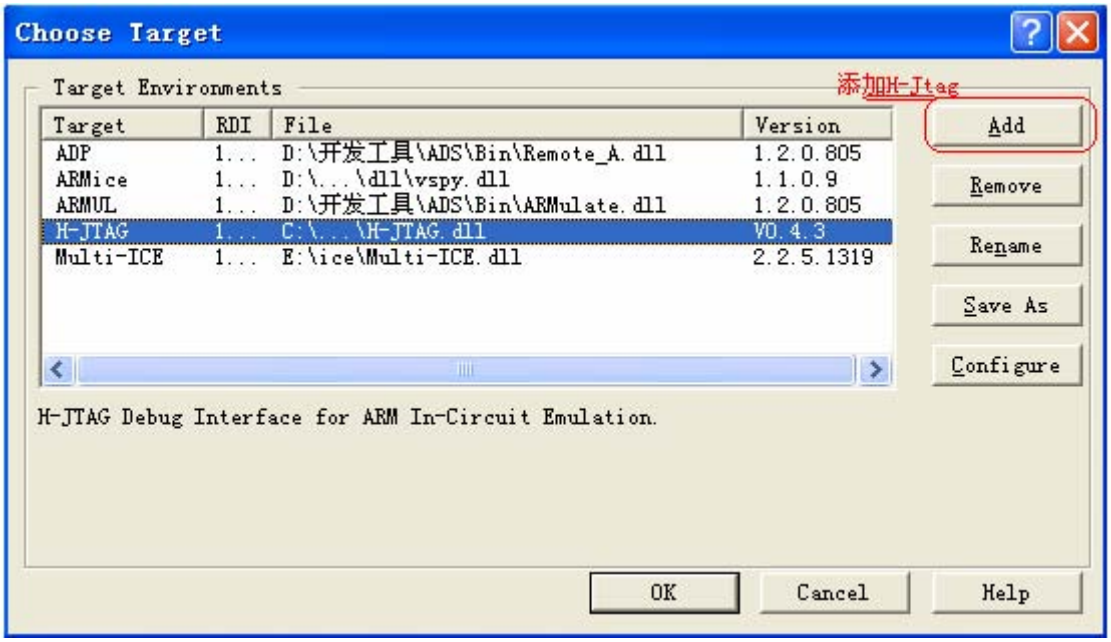

图 3－1 调试目标设置对话框

在 Target Environment 栏中选中“H-Jtag ”选项，注意如果没有 H-Jtag.all 文件，请添加。点击 Add 按钮，进入 H-JTAG 已安装目录找到 H-JTAG.dll 并打开。

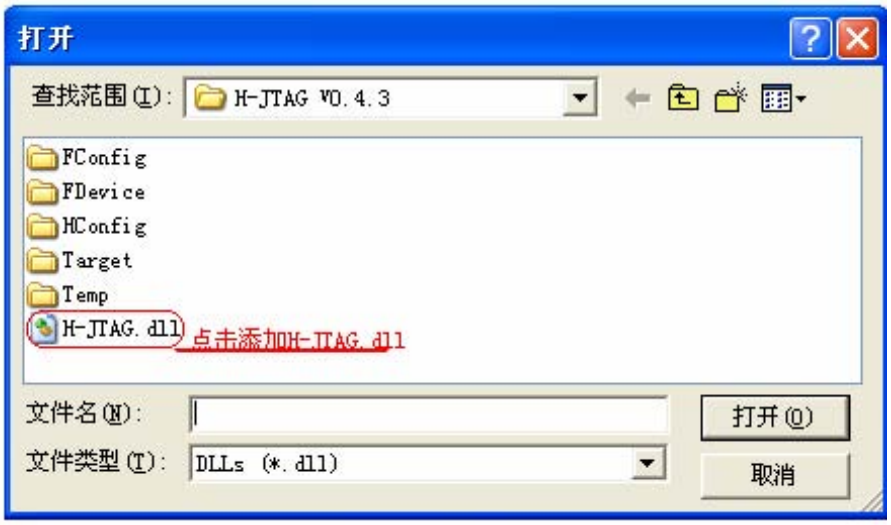

图 3－2 载入 H-JTAG 对话框

点击“是”按钮，如果目标系统正确链接了，会看到程序下载的进度条显示。进度消息框消失后，显示当前执行代码视窗，蓝色指针指向第一条执行的语句：

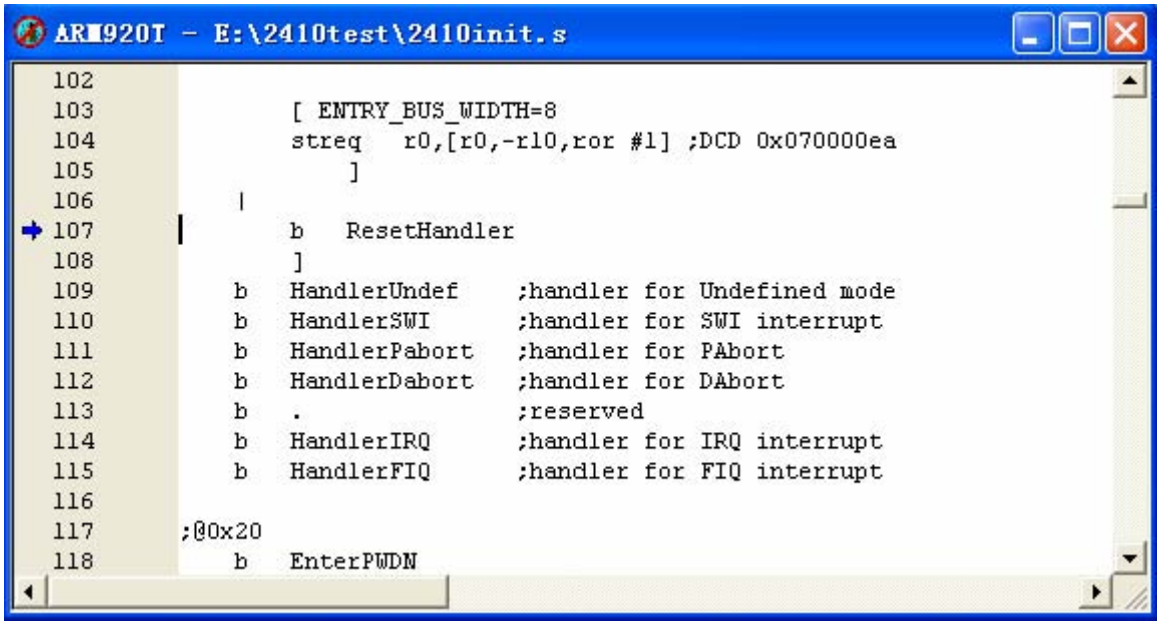

图 3－3 当前执行代码窗口

这时，先点击  按钮，尝试进行单步运行，如果程序立即正确地跳转到“ResetHandler”处执行，而没有跑飞或顺序执行，则说明程序的下载成功了，可以进行调试了。

3.3 AXD 调试器的使用

我们首先来熟悉一下断点的设置。下拉滚动条至 340 行，在 BL Main 语句处点击按钮  设置一个断点，如图 4-14。然后点击按钮  （GO），令程序自动执行到断点。当程序执行到 BL Main 语句处，自动停止，点击按钮  ，程序跳转到 2410test.c 文件的的 Main() 处程序开始运行，如图 4-15。

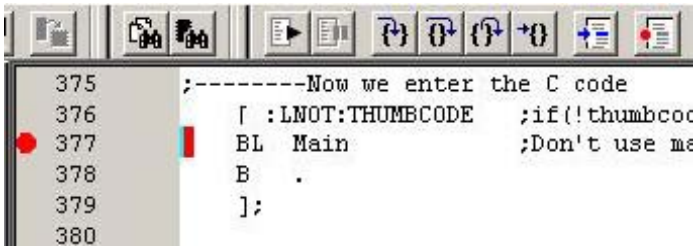

图 3－4 放置断点

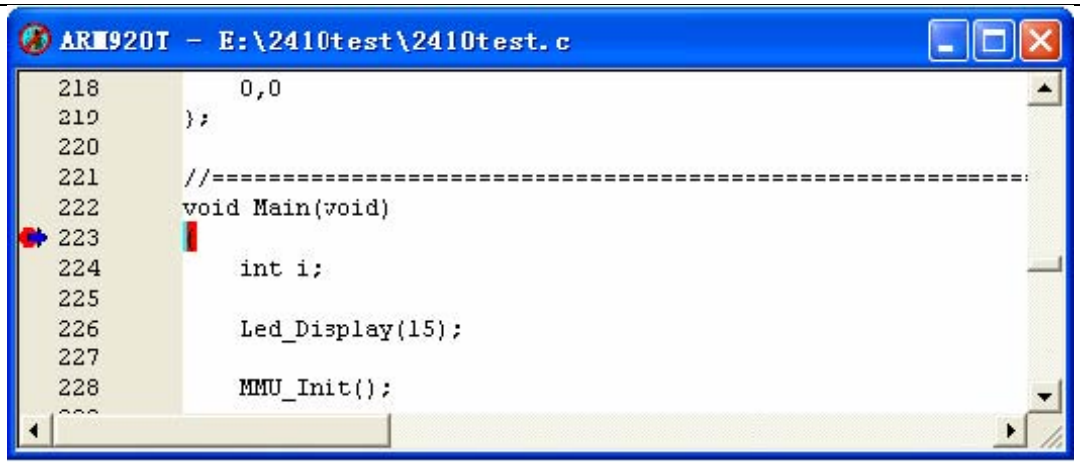

图 3－5 进入主函数运行

通过上面的操作，我们了解到，2410init.s 程序中的 BL Main 语句就是跳转到 C 语言 main（）函数的入口语句。AXD 也会自动在 Main()函数的入口处放置一个断点，因此程序下载后，立即全速运行的话，就会首先跳到该断点停下来。

读者可以继续进行一些单步操作，了解每条语句的作用。

3.4 AXD 观测窗口

AXD 提供了许多有用的观察窗口，点击菜单项中的 Processor View ，可以从它的下拉菜单项中了解可观察的项目。

图 3-6 观测窗口菜单

这里说明一下其中常用的项目：

Registers：可以查看 CPU 在各个工作模式下内部寄存器的值；

Variables ：查看变量，本地变量、全局变量、类变量；

Watch ：可以用表达式查看变量的值；

Backtrace：函数调用情况（堆栈）查看；

Memory：查看存储器内容。输入地址，即可查看这个地址开始的存储单元的值。

3.5 程序全速运行

在 AXD 中点击  ‘GO’图标，可以全速地运行程序，注意观察超级终端窗体，上面将显示如下信息。

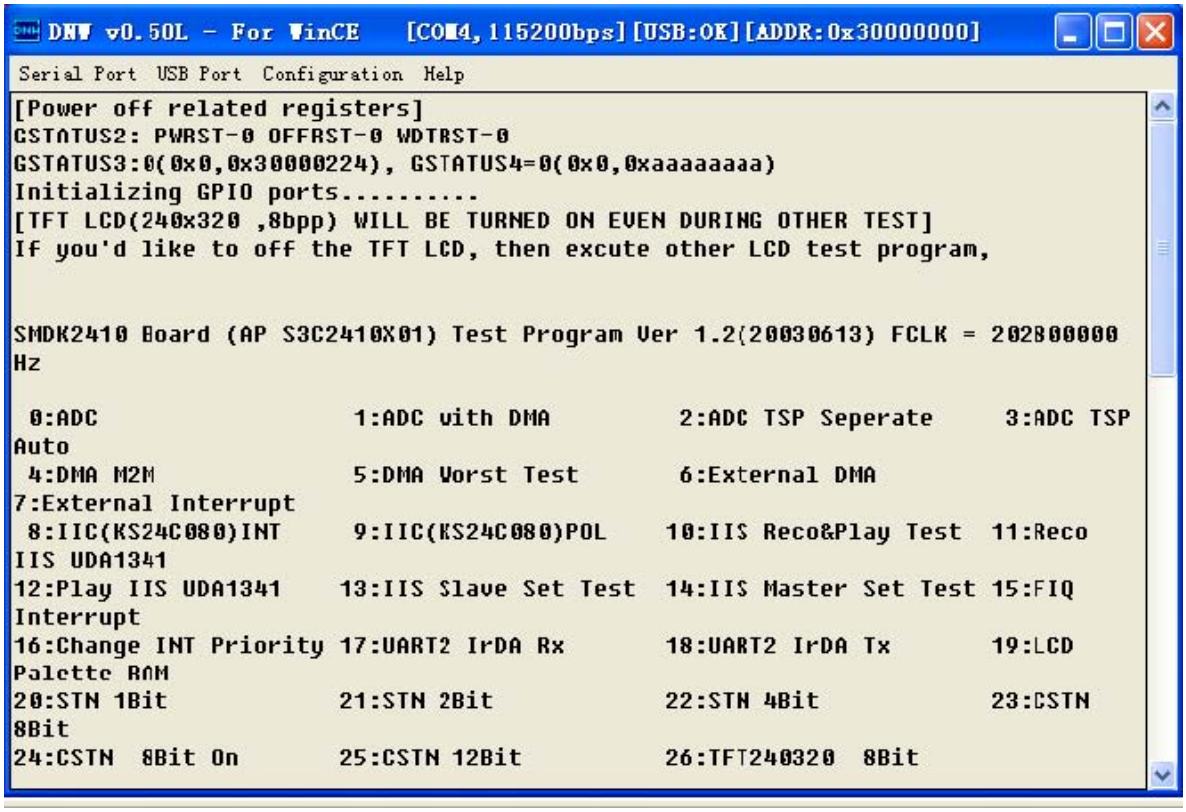

图 3－7 2410test 运行后超级终端的显示

在超级终端上显示硬件测试选择项后，在计算机键盘上键入相应的项目，超级终端上出现相应的测试结果了。