

一步一步基于 ADS1.2 进行设计开发

目 录

1 ADS1.2 集成开发环境简介.....	2
2 利用 HELLOWORLD 来学习使用 ARMSYS	2
3 编写好源程序代码	3
4 使用 CODEWARRIOR 建立工程并进行编译.....	4
4. 1 调入模板或重新建立项目	4
4. 2 在工程中添加源文件.....	8
4. 3 进行编译和链接.....	10
5 使用 AXD 进行仿真调试.....	11
5. 1 硬件准备.....	11
5. 2 使用 UART 串口和超级终端进行系统调试.....	11
5. 3 运行 JTAG 调试代理软件.....	12
5. 4 调试器设置.....	13
5. 5 调试器的使用.....	15
5. 6 观察窗口.....	16
5. 7 全速运行.....	17
6 USB 口下载工具.....	17
7 代码固化	21
7. 1 空板烧录.....	22
7. 2 FLASH 内代码的覆盖烧录.....	22

1 ADS1.2 集成开发环境简介

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：

- n 命令行开发工具；
- n 图形界面开发工具；
- n 各种辅助工具；
- n 支持软件。

其中重点介绍一下图形界面开发工具。

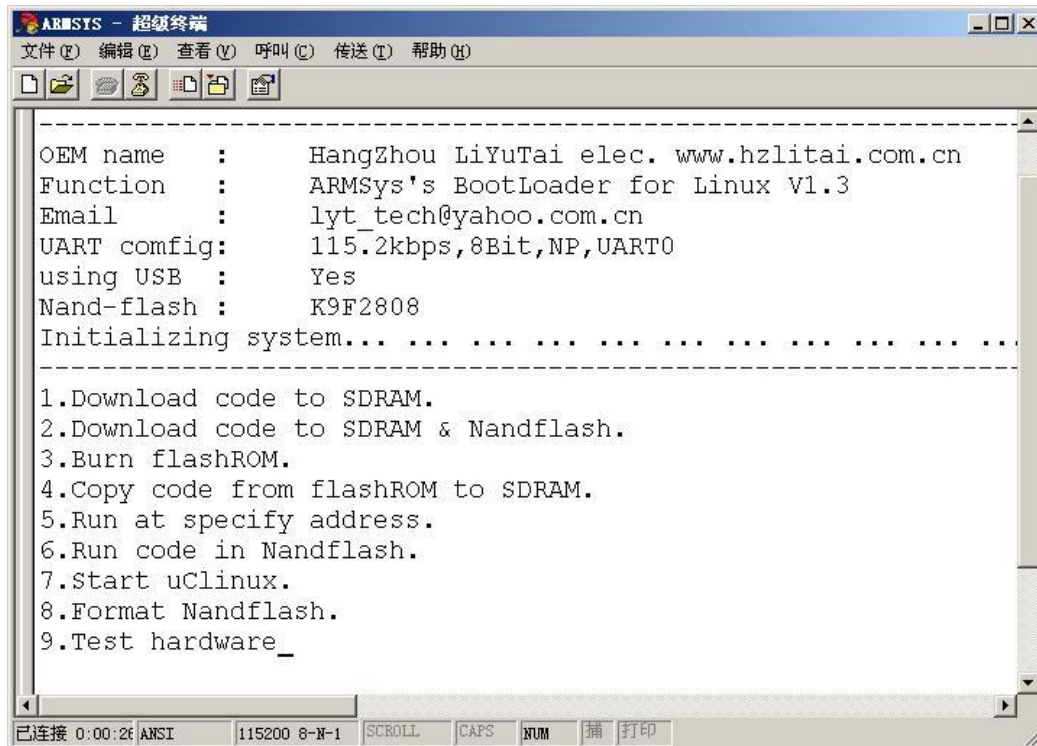
- n **AXD** 提供给基于 Windows 和 UNIX 使用的 ARM 调试器。它提供了一个完全的 Windows 和 UNIX 环境来调试你的 C, C++, 和汇编语言级的代码。
- n **CodeWarrior IDE** 提供基于 Windows 使用的工程管理工具。它的使用使源码文件的管理和编译工程变得非常方便。但 CodeWarrior IDE 在 UNIX 下不能使用。

2 利用 Helloworld 来学习使用 ARMSYS

本实验利用光盘中 source\Helloworld\下的源代码进行实验，要求完成以下工作：

- 1) 利用工程模板 source\template.mcp，在 ADS1.2 的 **CodeWarrior IDE**（项目管理器）中建立新的工程，在工程中加入 source\Helloworld\Target 目录中提供的文件；
 - 2) 编写主程序文件 main.c，并将文件加入到工程中；
 - 3) 正确设置编译器的编译选项，并对工程进行编译、除错，最终产生可执行的映像文件（*.axf）和二进制代码(*.bin)文件；
 - 4) 打开超级终端，正确配置串口参数；
 - 5) 采用 AXD（视窗调试器）通过 JTAG 模块下载可执行程序，并仿真调试，观察实验现象；
 - 6) 采用 ARMSys 提供的 USB 下载器下载二进制代码，并观察运行情况。
- 将 source 目录整个拷贝到硬盘中，例如拷贝到 D:\source 处，方便进行实验。

【注意】在进行 ADS 程序调试之前，确保开发板中已经固化了 bootloader.bin。Bootloader 启动时超级终端上应当显示如下：



如果板之上已固化的不是 **bootloader**，请按照 7.2 节中的说明，讲 **bootloader** 固化到 **flash** 中。

首先，安装 ADS1.2，在光盘的\开发工具\集成开发环境\Windows\目录下有 ADS1.2 的安装文件，双击 setup.exe 进行安装，具体的安装过程这里就不赘述了。

3 编写好源程序代码

用 **CodeWarrior IDE** 打开 source\HelloWorld\main.C 文件，读懂其中的代码。其中 main() 函数的定义是：

```
void Main(void)
{
    char aa;

    Uart_Init(0,115200); //初始化 UART0 口，设置波特率为 115200bps
    Led_Display(0xf);   //点亮绿色发光二极管
    Uart_Select(0);      //选中 UART0
    Beep(0x1);           //点响蜂鸣器
    Uart_Printf("\n*****");
    //向串口输出字符串
    Beep(0x0); //
    Uart_Printf("\n*      立字泰电子      *");
    Uart_Printf("\n*      -Hello World!-      *");
    Uart_Printf("\n*      Version 1.21      *");
    Uart_Printf("\n*      Email:Support@hzlitai.com.cn      *");
}
```

```

    Uart_Printf("\n*   UART Config--COM:115.2kbps,8Bit,NP,UART0 *");
    Uart_Printf("\n*Begin to Study Embedded System,OK?(Y/N)--- *");
    Led_Display(0x0);
    aa = Uart_Getch();//等待并从串口获得一个字符
    if((aa=='Y')||(aa=='y'))
    Uart_Printf("\nGood!See you next time!");
    else
    Uart_Printf("\nByeBye!");
}

```

这是一个简单的例子，因此代码比较简单。这里你可以尝试修改一下代码，将：

```
Uart_Printf("\nGood!See you next time!");
```

修改为：

```
Uart_Printf("\n 好！我们马上开始学习嵌入式系统！");
```

然后保存文件。

除了 main.C 文件以外，在一个工程中，我们还要具备几个必不可少的源文件：系统初始化程序 44binit.s，和它引用的存储器控制寄存器定义文件 Memcfg.s 和系统选项文件 Option.s，这 3 个文件都由汇编语言写成。C 语言基本函数库定义文件 44blib.a（或 44blib.c，[该文件可以在/BIOS/Target 或者 bootloader/Target 下找到](#)）和头文件 44blib.h、option.h、44b.h。这些文件定义了最基本硬件系统信息，几乎在所有的应用工程中都要使用到，你可以在每个工程的 Target 目录下找到它们。

4 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（例如汇编或 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。

4. 1 调入模板或重新建立项目

我们通常采用工程模板来建立新的工程，工程模板已经针对目标系统对编译选项进行了设置，为避免重复设置，我们提供了一个在 ARMSys 上使用的通用工程模板——template.mcp。

点击 CODEWARRIOR 菜单[File | open...], 找到 source\template.mcp, 选中并打开。

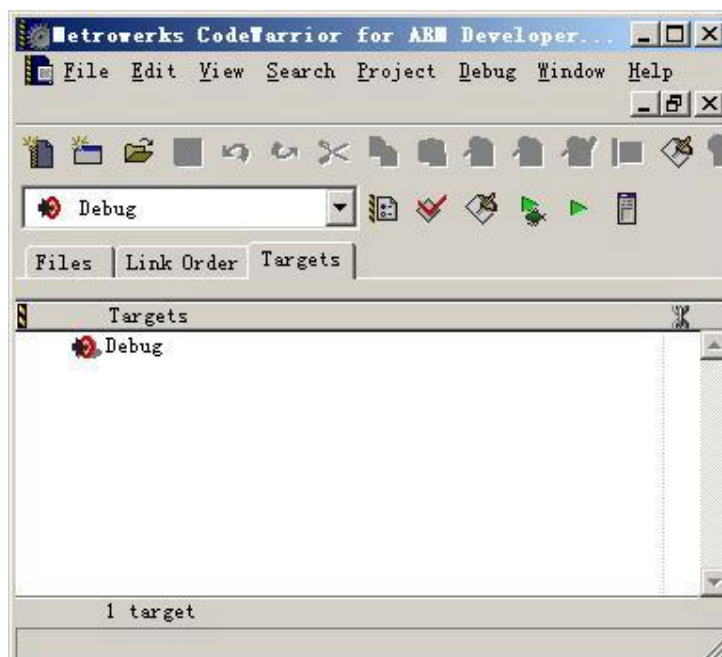



图 6 打开模板工程

点击[File | Save as...], 将它另存为: source\Myhelloworld (或者是自定义的其它目录) \ Myhelloworld.mcp。然后, 关闭当前的工程, 重新调入 Myhelloworld.mcp, 就可以向工程中添加文件了。

如果你不想利用模板, 也可以按照以下步骤来新建一个工程:

选择 File 菜单下的 new 选项, 或直接按下 , 出现以下对话框:

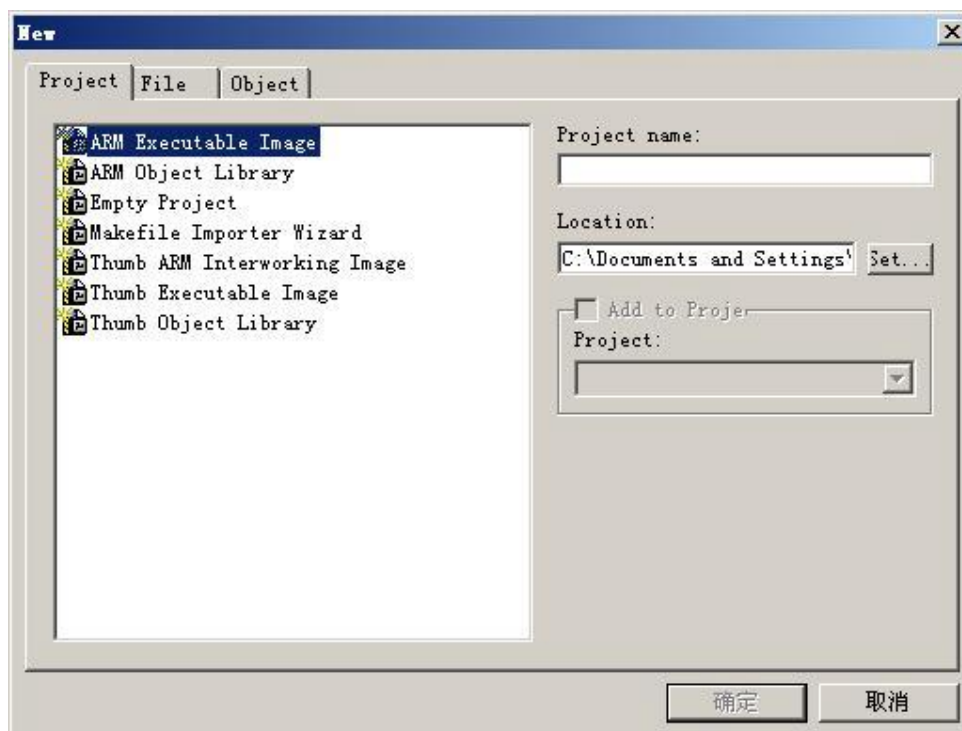


图 7 新建对话框

- 2). 选中“ARM Executable Image”选项, 在右边的编辑框中输入工程名(例如 Myhelloworld), 在下面的 Location 栏中, 点击 “Set...”, 选择放置工程的路径。
- 3). 点击 “确定” 则工程被建立:

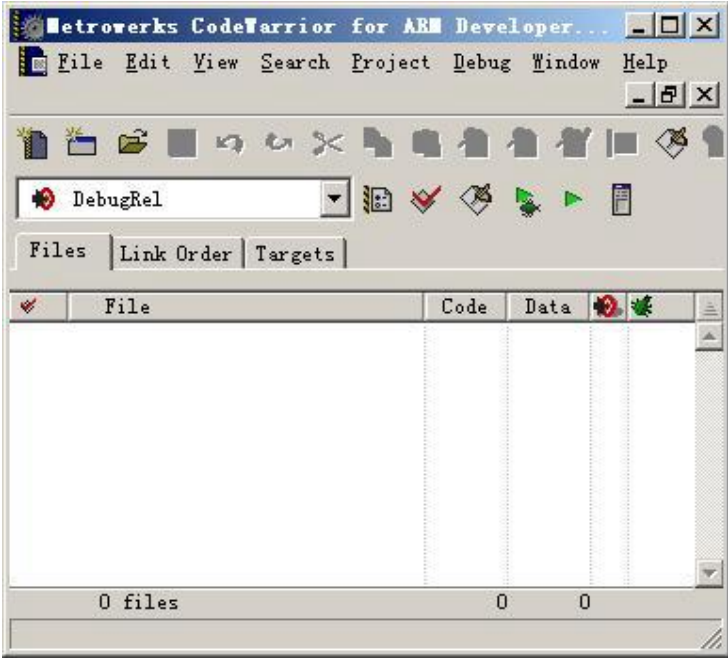


图 8 新建工程

但这样的工程还并不能正确地编译, 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 Release 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:

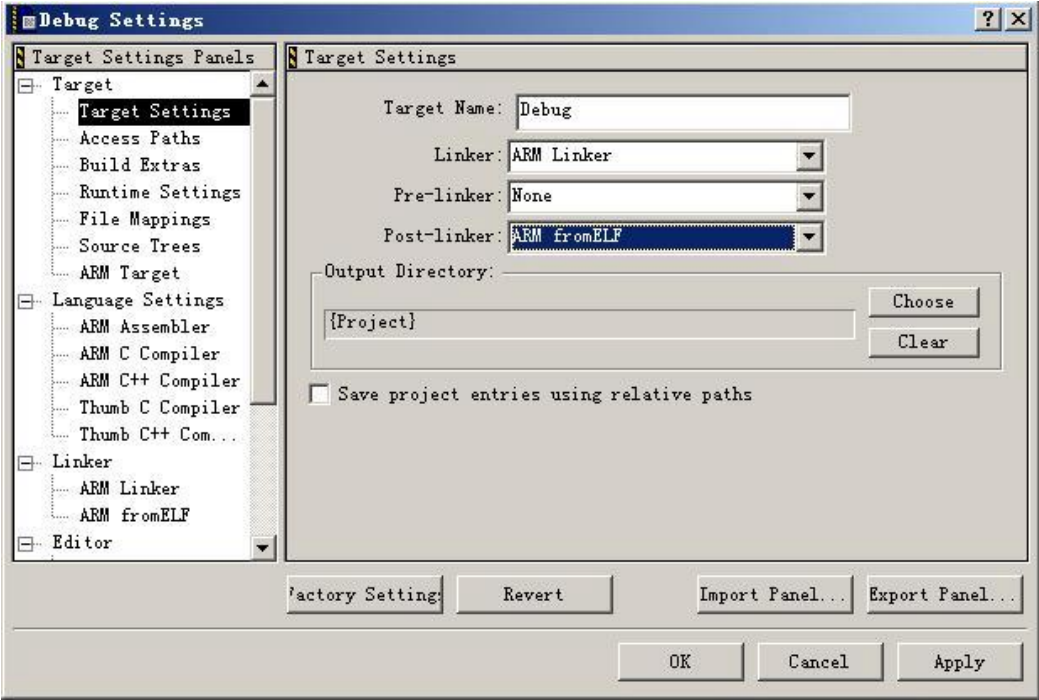


图 9 工程配置对话框——目标设置

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：

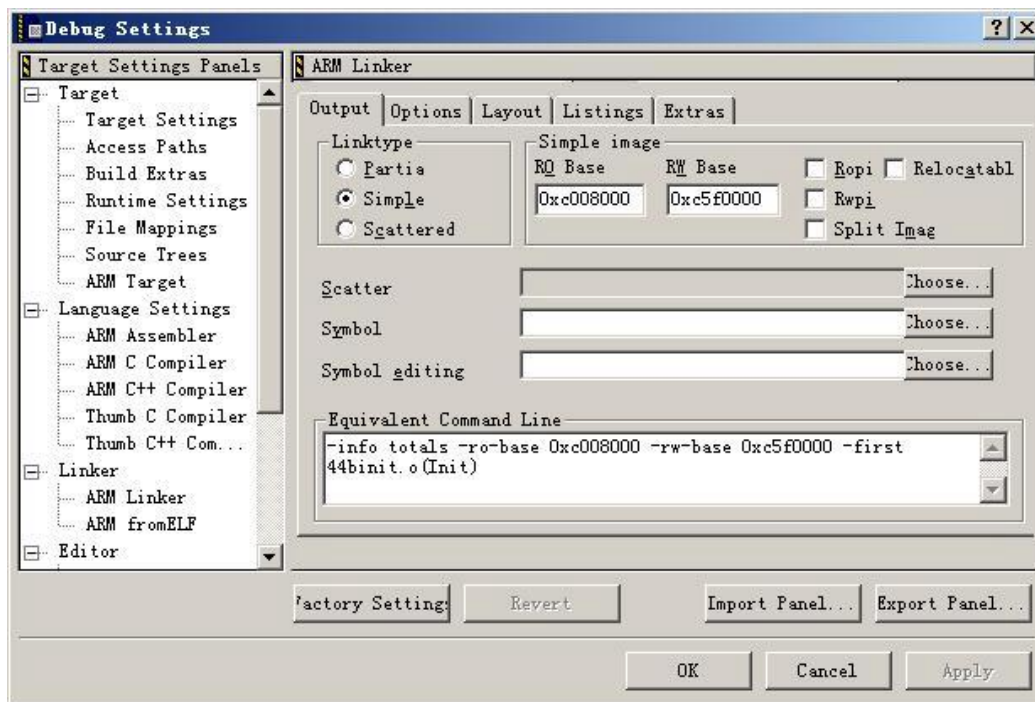


图 10 (a) ARM Linker 的设置

注意，在调试时，-ro-base 的设置应当大于 0xc000000。我们为了与 uClinux 的 memory map 保持一致，采用了 0xc008000 这个地址。

选取 Layout 页面进行设置：

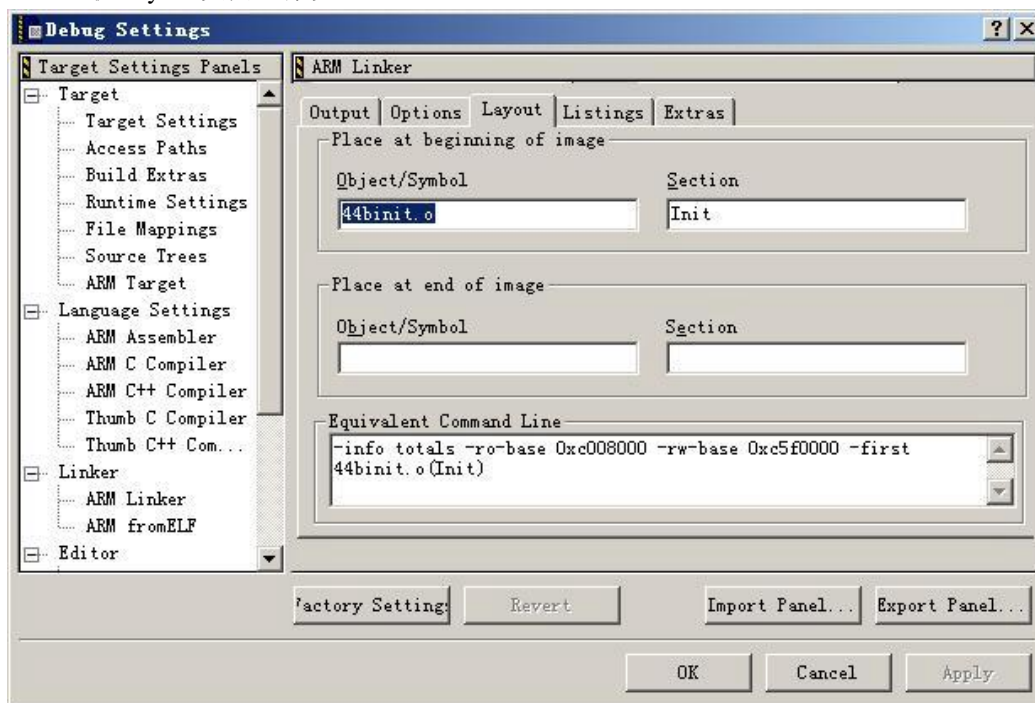


图 10 (b) ARM Linker 的设置

将 44binit.o 放在映像文件的最前面，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：

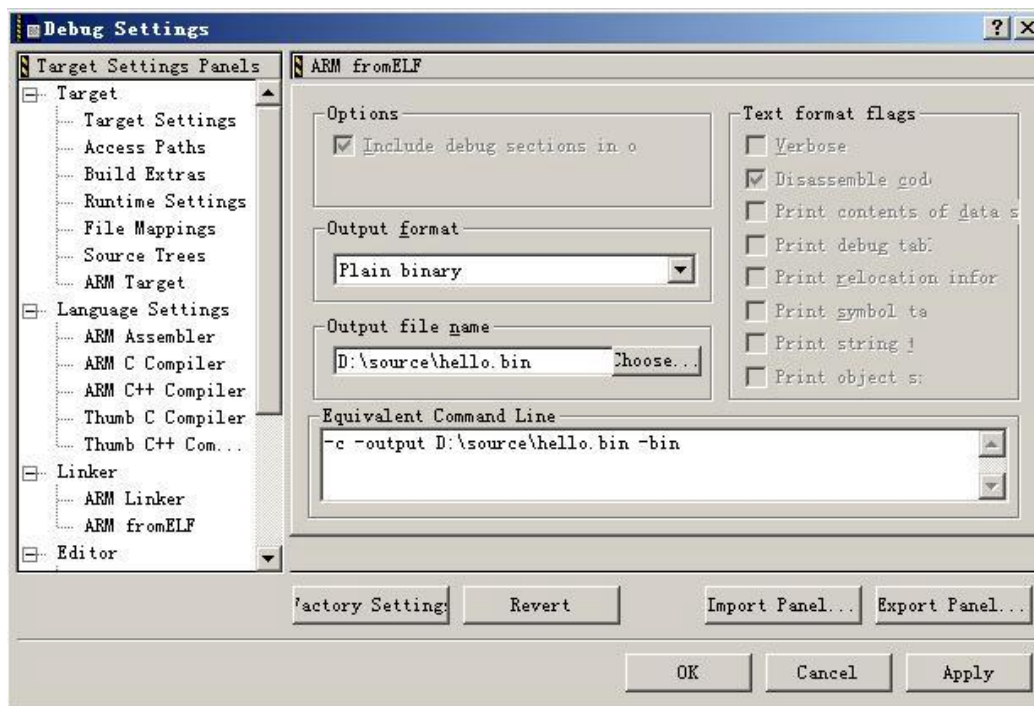


图 11 ARM fromELF 的设置

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（如果此栏为空，则二进制文件将会产生到默认的工程目录下）。

这样，对于 Debug 变量的基本设置都完成了。按下“OK”键退出。

4. 2 在工程中添加源文件

在图 7 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 main.c 文件，点选 Files 页面，在空白处按下鼠标右键，点选“Add Files”项，从目录中选取 main.c 文件（Myhelloworld\main.c），点击“打开”，main.c 文件就被加入了工程中。

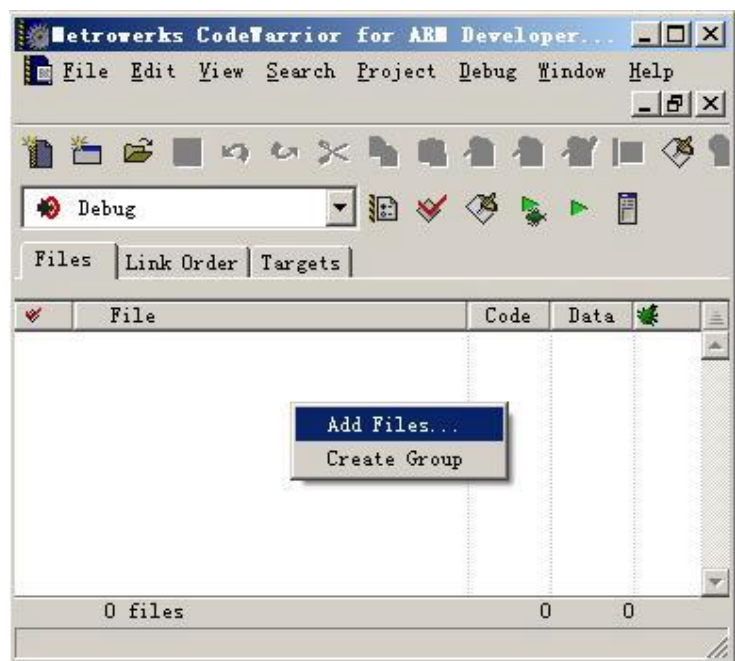


图 12 添加源文件

用同样的方法，将 Myhelloworld\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)。

Target 目录下还有一个 44b.lib.a 文件，这是一个库文件，其中提供了一些常用函数的定义，这些函数在 44b.lib.h 进行了声明。这个文件也必须添加到工程中。同样的方法，按鼠标右键，Add files...，将 44b.lib.a 文件添加到工程中。所有必须的文件添加完成后如图 13 所示。

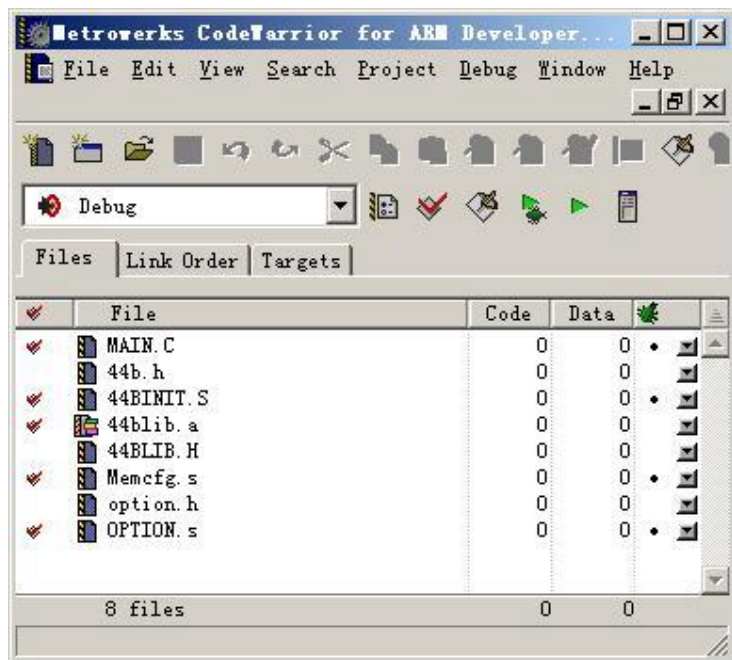





图 13 源文件添加完成

4. 3 进行编译和链接

注意到在上图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 4.1 节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。注意到左上角标示的错误和警告数目都是 0：

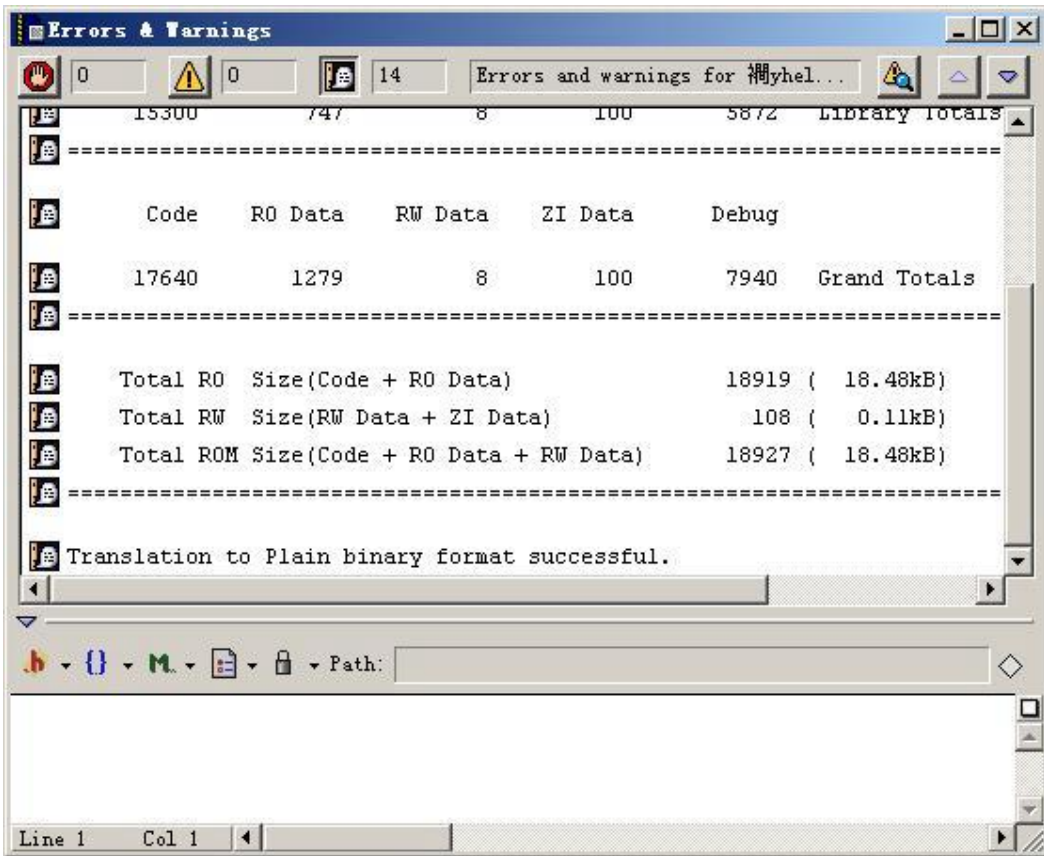


图 14 编译后的结果

Make 结束后产生了可执行映像文件 Myhelloworld.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 `hello.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），也可以通过 USB 口下载运行。

5 使用 AXD 进行仿真调试

5.1 硬件准备

如图 4 所示，在调试之前，我们先用并口电缆将 PC 机并口和 JTAG 调试模块连接起来，用串口线将 PC 机串口和主板的 UART0 口连接起来（当然还要将主板和 JTAG 板连接起来）。然后，就可以上电了。参考 2.4 节选择 ARMSYS 的供电方式。

电源打开之后，可以听到主板发出一声蜂鸣器的“嘀——”声，看到绿色发光管点亮后熄灭，这说明主板启动正常。此时 JTAG 模块上只有指示电源的红灯点亮，说明并口已经连接好了。

5.2 使用 UART 串口和超级终端进行系统调试

在 Windows 操作系统下，点击[开始 | 程序 | 附件|通讯 | 超级终端]。新建一个超级终端项目，将其命名为 ARMSYS，点击“确定”，弹出以下对话框：

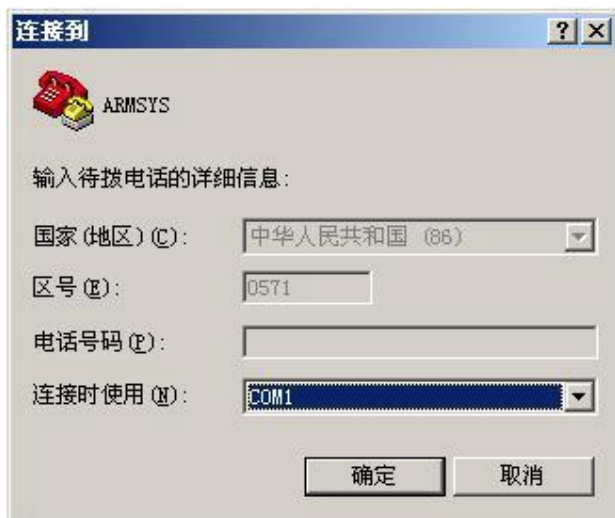


图 15 超级终端属性

在“连接时使用”项中选好你所使用的串口，点击“确定”按钮。按照下图配置该串口：



图 16 串口属性配置

点击确定，超级终端就配置好了。

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 JTAG 调试代理软件。如果采用 Multi-ICE 仿真器来调试，则首先要运行 Multi-ICE Server（具体请查看仿真器的使用说明）。

5. 3 运行 JTAG 调试代理软件


将光盘中的开发工具\ARMJtagDebugFinal\目录拷贝至硬盘某个目录下，按照其中《使用说明》安装驱动程序。驱动安装成功后，双击 ARM7.exe 运行调试代理软件。如果调试代理软件与目标系统连接成功，则显示以下对话框：



图 17 JTAG 调试代理软件运行对话框

在运行 AXD 调试器之前必须首先运行它。注意，在 AXD 调试器在线仿真期间，不要关闭它！

5. 4 调试器设置

在 CODEWARRIOR 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：

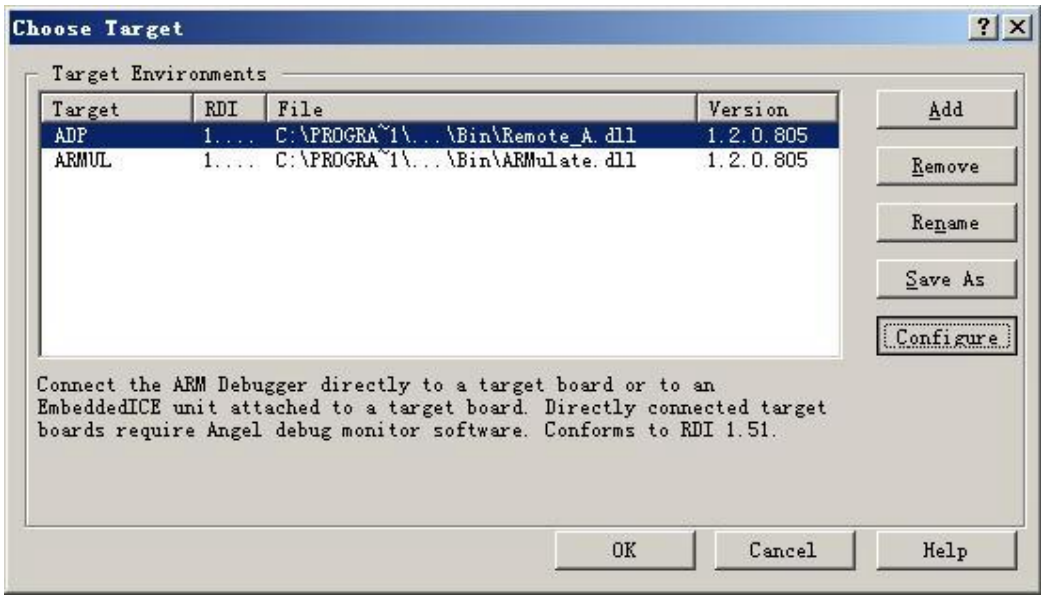


图 18 调试目标设置对话框

在 Target Environment 栏中选中“ADP”选项，注意到下面的注释的说明，“ADP 是直接连接 ARM Debugger 到目标板或者到目标板上的 EmbeddedICE 单元的一种方式。直接连接目标板需要 Angel 调试监视器软件的支持。参考 RDI 1.51”，点击“Configure”按钮，进入到该项设置：

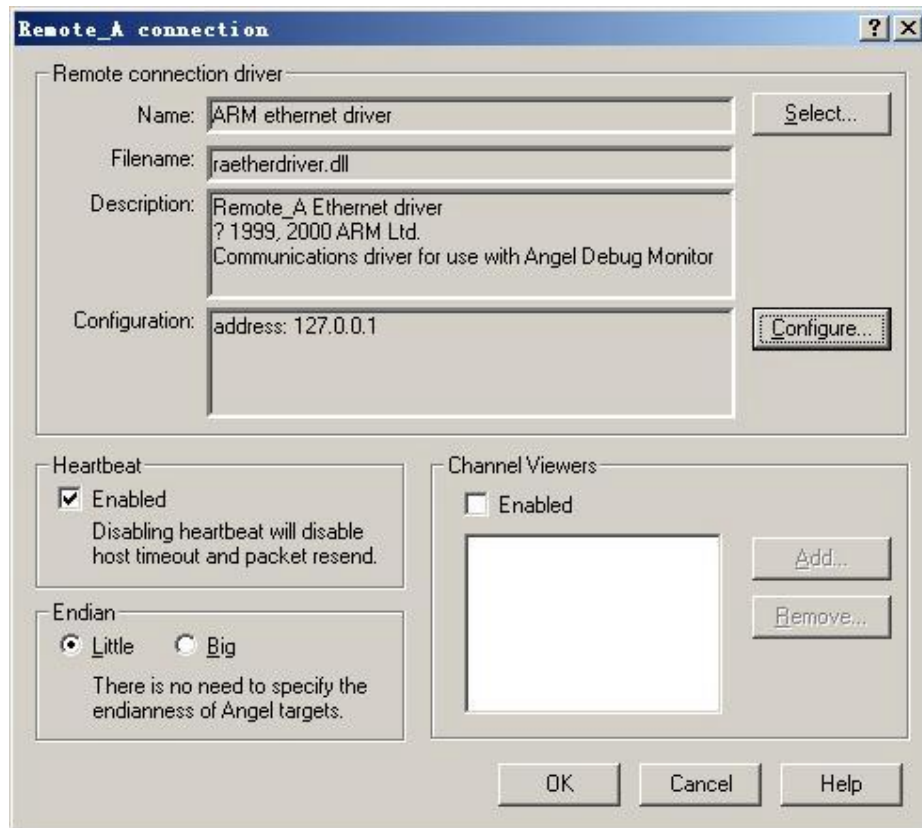


图 19 Remote_A 连接设置

其中“Remote Connection driver”栏中，点击右边的“Select...”按钮，选择“ARM ethernet driver”。点击右边的“Configure...”按钮，在编辑栏中输入本机的 IP 地址或者 127.0.0.1。其它设置如上图所示，保持不变。点击“OK”退出调试目标的设置。这时会弹出：

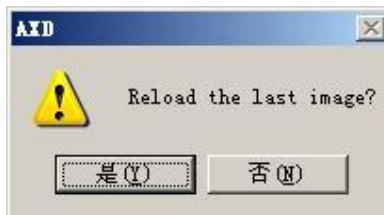


图 20 重新载入对话框

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：

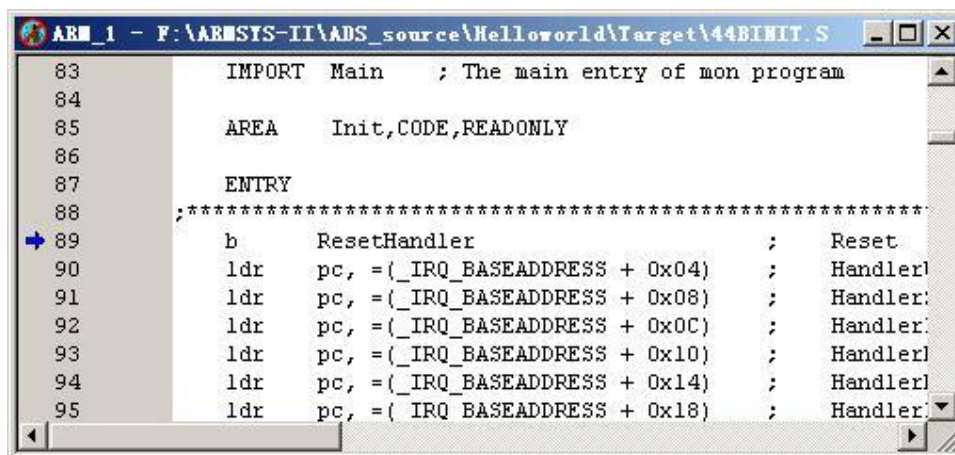






图 21 当前执行代码视窗

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。

5. 5 调试器的使用

我们来熟悉一下断点的设置。下拉滚动条至 377 行，在 BL Main 语句处点击按钮  设置一个断点，如图 22。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 main.c 文件的 Main() 处程序开始运行，如图 23。

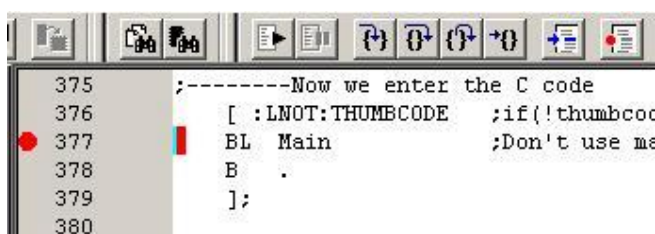


图 22 放置断点

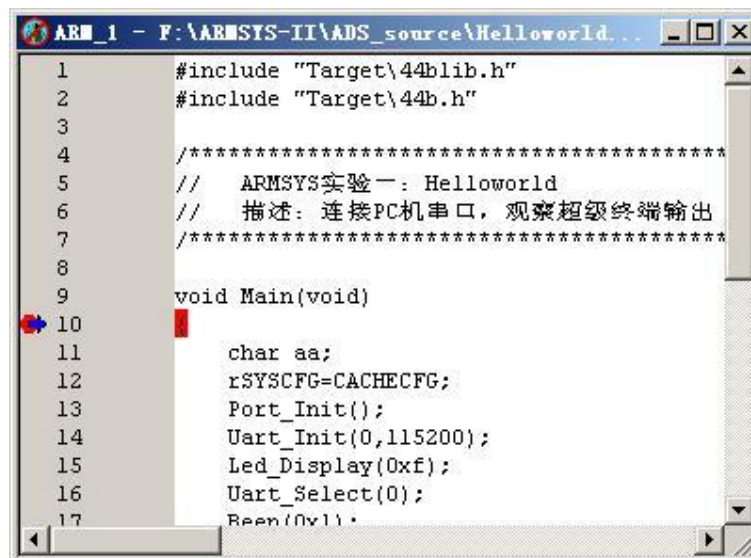


图 23 进入主函数运行

通过上面的操作，我们了解到，44binit.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。

5. 6 观察窗口

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。

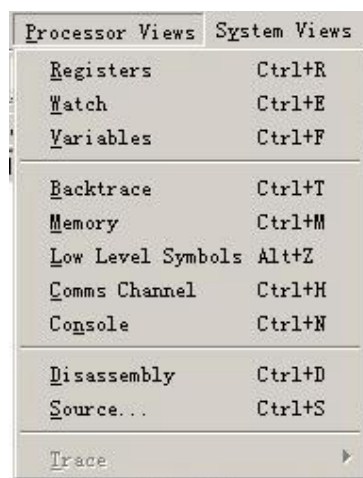


图 24 观察窗口

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。

.....

5. 7 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。

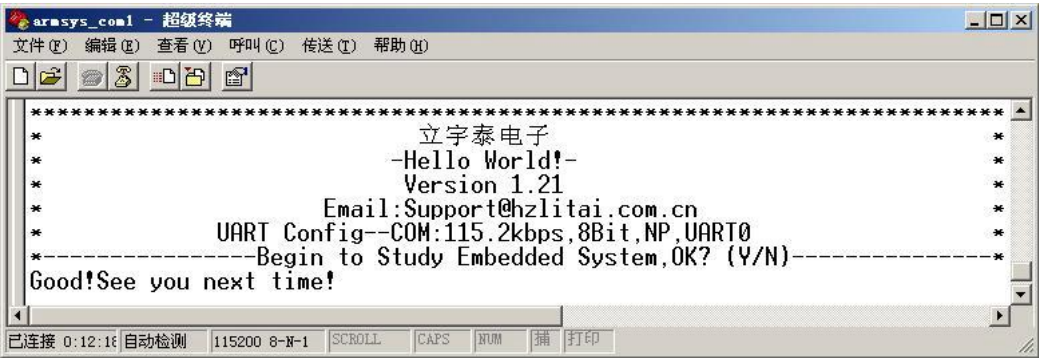


图 25 HelloWorld 运行后超级终端的显示

在 “Begin to Study Embedded System, OK? (Y/N)” 后，在计算机键盘上键入 Y，超级终端上出现 “Good! See you next time!”。如果你按照第 3 节修改了源程序，则应当出现 “好！我们马上开始学习嵌入式系统！” 字符串。

6 USB 口下载工具

USB 口下载工具能够将二进制代码快速下载到 ARMSys 上并运行。用户能够立即观察到程序运行的效果。USB 下载器工具放在光盘的开发工具\usb 目录下。安装驱动程序步骤如下：

- 1 步骤 1，将 usbinstall 目录，整个拷贝到 C:\下（注意必须是 C:\）；
- 1 步骤 2，双击 C:\usbinstall\install.bat，等待一会儿，直到整个批处理程序执行完毕（注意：弹出的控制台窗口自动消失之前不要手动关闭！在 98 下标题栏中最后显示 “已完成”）；
- 1 步骤 3，用 USB 口线连接 ARMSys 核心板和 PC 机 USB 口，打开电源；开发板正确复位后超级终端显示如下：

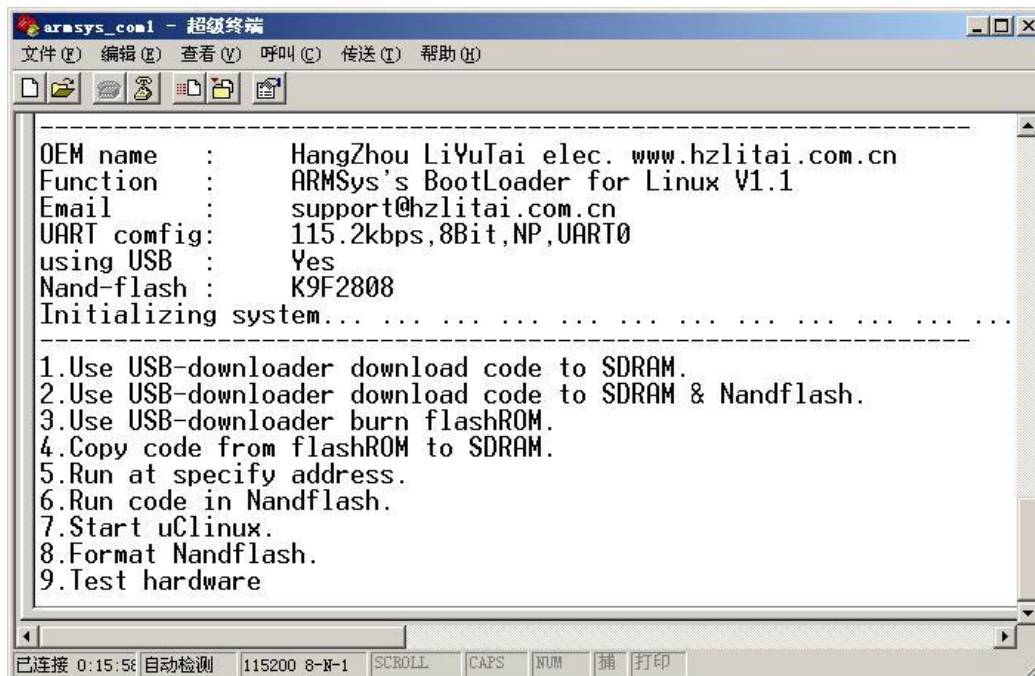


图 26 复位时超级终端的输出

按下键盘上的‘1’键，会提示你输入地址，如下图所示：

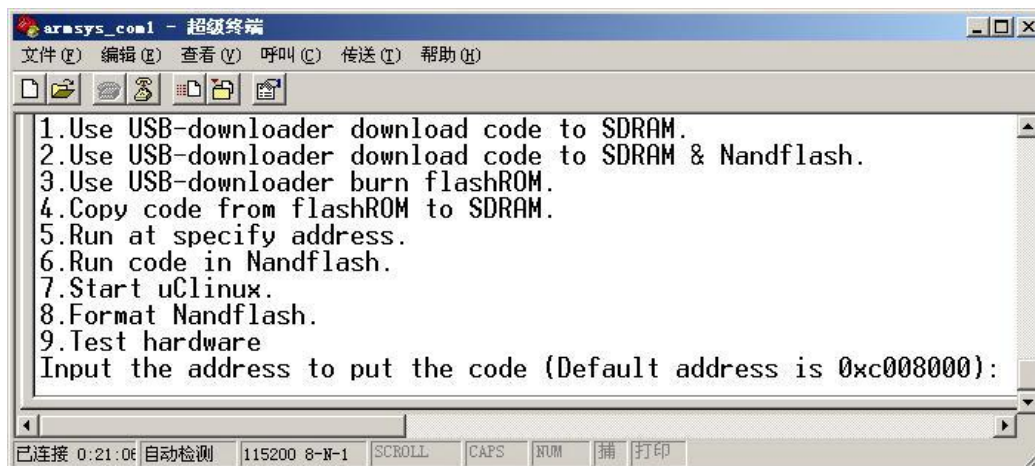


图 27 输入 SDRAM 载入地址

这个地址就是你下载到 SDRAM 中的地址，也是链接器代码定位的地址，直接敲回车则使用括号中的缺省地址。输入地址并回车后出现：

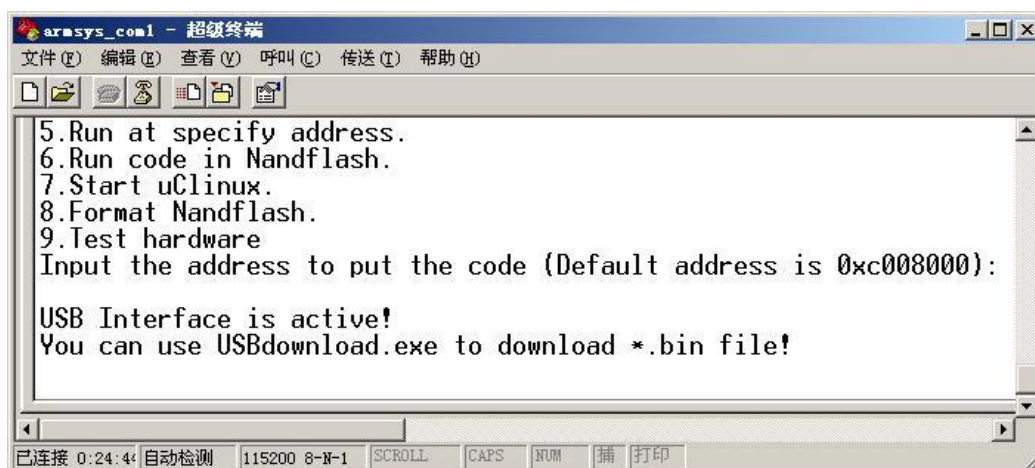


图 28 USB 设备被激活

这样 USB 接口就成功地被使能了；这时可能会报告“找到新硬件”，一般可以让驱动指向 usbinstall 目录下即可。打开“设备管理器”会看到 Jungo 下面自动识别出了 USB Tool Device 设备，说明驱动安装成功了。



图 29 设备管理器树形图

I 步骤 4，双击运行 C:\usbinstall\Usbdownload.exe，出现如下对话框界面：

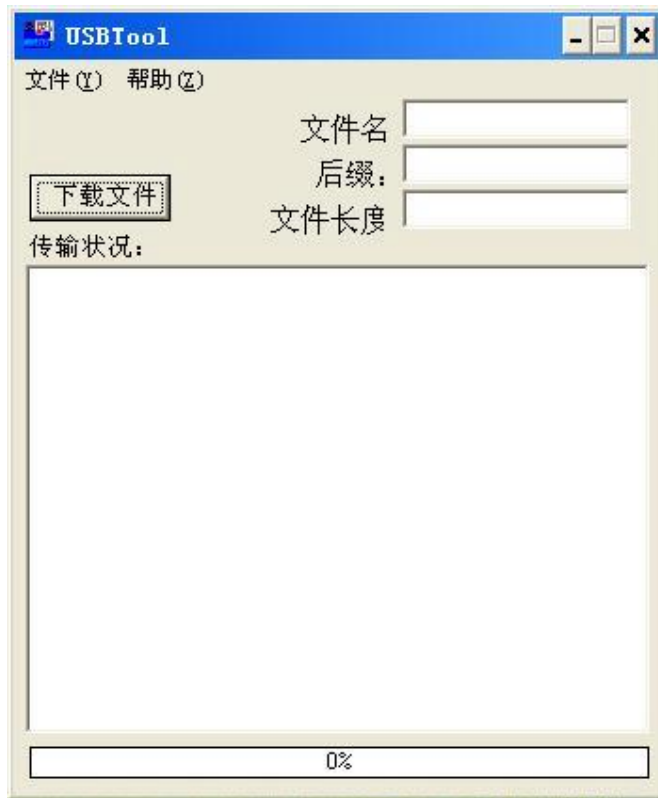


图 26 USB 下载工具界面

- I 步骤 6，点击 USBTool 工具的菜单项[文件 | 打开文件]，然后在“文件类型”中下拉选择*.bin 类型文件，选择好要下载的文件后（例如刚刚产生的 `helloworld.bin` 文件），点击“打开”。这时可以看到，对话框的文件名、后缀、文件长度编辑框中即出现所选文件的相关信息。点击“下载文件”按钮，二进制文件被下载到 ARMSys 中。等待一会儿，系统自动开始运行刚刚下载的程序。通过超级终端上显示的内容，可以看到程序的运行情况。

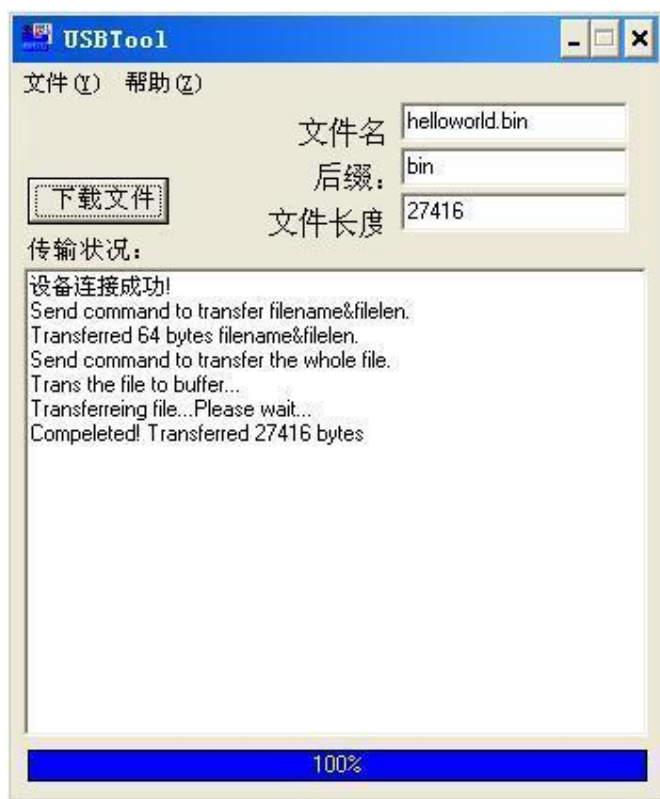


图 27 下载成功界面

7 代码固化

请注意，将工程编译为烧入二进制代码时，需要将链接器重新设置，将调试时程序空间定位地址（RO Base 地址）由 0x0c008000 改为 0x00000000，如下图所示。

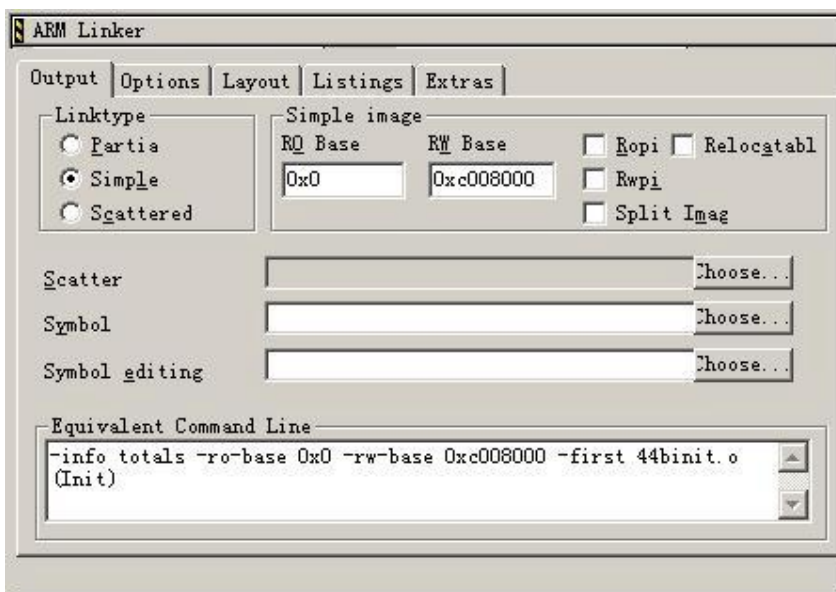


图 28 Entry and Base 设置页

然后重新 **Make** 整个工程。产生新的二进制代码文件。

7. 1 空板烧录

注意，以下说明的是空板烧录。如果您的开发板中已经烧录过启动程序，并且能够正常运行，则不必进行这一步，直接跳到 7.2 节。

建议采用 **fluted** 工具，由于 **fluted** 烧写速度较慢，可以先烧录一个 **led_test.bin** 小程序，只要烧录的程序在启动后能够正确初始化 **SDRAM** 即可，具体方法参考光盘开发工具 **/programmer/fluted** 目录中的说明；然后再进行 7.2，烧录你的应用程序代码。

7. 2 FLASH 内代码的覆盖烧录

采用光盘开发工具 **/programmer/ADS_programmer** 目录下提供的工程进行烧录。按照第 4、5 节的方法，对该工程进行编译然后载入开发板进行调试。

载入成功后，点击菜单 **File > Load Memory From File...**，弹出图 29 的对话框，在 **Address** 中写入 **0xc200000**，选中你要烧录的 **bin** 文件，点击打开。

这样 **bin** 文件很快被下载到相应的地址。全速运行烧录程序。可以通过超级终端观察到烧录的情况，烧录过程分为片擦除、空检查、写入和校验，烧录成功的输出信息为：

Chip erased!

Blank check OK! Begin to Write flash...

Write OK! Begin to Verify...

Verify OK!

如果终端输出以上信息，其间没有报告错误，说明烧录已经成功了。

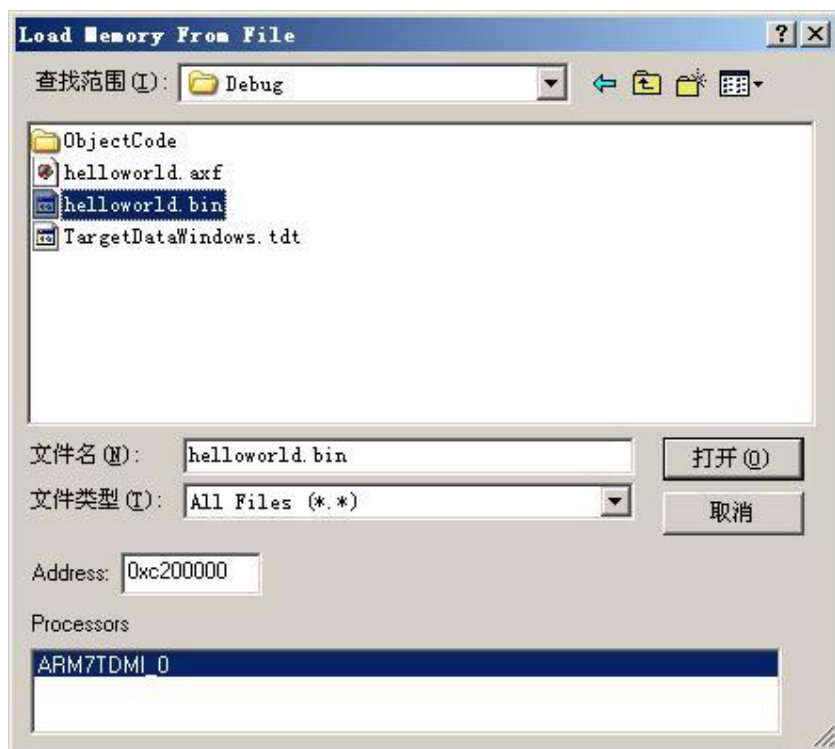


图 29 Load Memory From File 对话框

到此为止，我们已经对 ARMSYS 上的整个开发过程进行了一次体验。读者可以按照以上步骤说明，多操作几次，有任何问题，欢迎来信与我公司的技术支持进行交流：office@hzlitai.com.cn、support@hzlitai.com.cn，或垂询技术支持电话：0571—88331446 89902166。