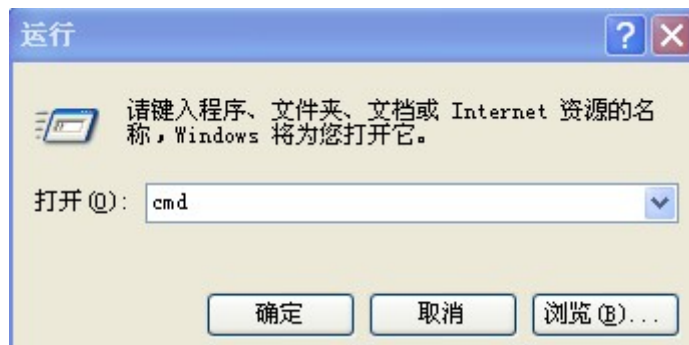


运行与调试

本文档适用于 44b0、2410、2440 的调试，其他 ARM 板子也可以参考。

1 编译操作系统

编译之前，请确保你已经建立 gcc 编译环境，可以按照《建立 windows 下 djyos for arm 的编译和调试环境.doc》一文建立 windows 下的 gcc 编译环境。在 windows 下，点“开始-运行”，在运行对话框：



中输入 cmd，点“确定”进入命令行方式，然后进入源程序所在目录（makefile 文件也在这个目录中），**注意，不是进入安装 cygwin 时带的“Cygwin Bash Shell”环境**。在这个目录中执行 make 命令即可编译。这里简单介绍一下 makefile 支持的几个编译命令。

1、make boot_rom 命令

编译产生 boot_rom.bin 文件，当需要用硬件仿真器调试操作系统时，先把 boot_rom.bin 烧录到 0 地址的 norflash 中。

2、make debug 命令

编译产生产生用于调试的 debug.elf 文件，编译器的优化级别是 0，且含 gdwarf-2 格式的调试信息，由调试软件比如 realview2.2 或 gdb 直接加载到内存中调试。

3、make run_inram 命令

编译产生二进制可执行文件 run_inram.bin，烧录到 0 地址的 norflash 中，复位启动后会自动把代码 copy 到 ram 中执行，si 版本的 djyos 没有独立的 bootloader，可近似理解为 si 版本内含 bootloader 的部分功能。

4、make clean 命令

删除除 .bin 和 .elf 外的所有由编译产生的文件。

源代码已经充分考虑到编译器的相关性，所以 djyos 可以用任意级别的编译器优化，所以你看到的 release 版本的代码量会比 debug 版本小很多，只有 55% 左右，gcc 的优化效果还是很惊人的。

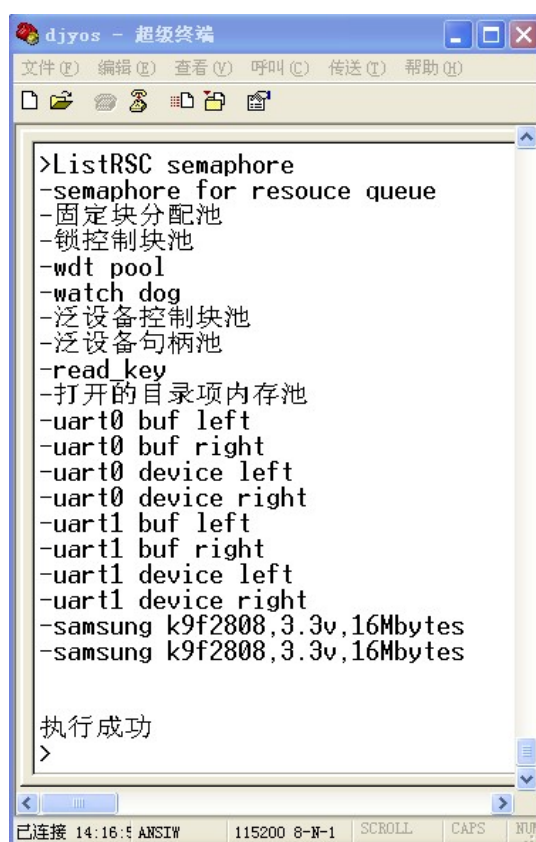
djyos 运行时需要多少 ram 基本上由 config.h 文件中的配置决定，如果不支持文件系统，再在 config.h 中使用比较小的配置的话，ram 需求可以降到 5Kbytes 以下，这意味着一颗只有 8Kram 的单片机也可以从容运行 djyos。

2 直接运行

把上面编译的 `run_inram.bin` 烧录到 flash 的 0 地址后，复位就可以运行，为确保成功复位，最好先拔掉 jtag 调试电缆。

可以有多种烧录工具可以把文件烧录到 flash，共享的文件包中提供了 jlink 的烧录工具 JFlashARM 的配置文件 `download.jflash`。具体烧录方法请参考所使用的烧录工具的说明文档，与本文档同时发布的有一个《用 jlink 烧录 flash.doc》文档。

连接开发板的 uart1 和计算机串口，打开超级终端，连接串口，设置为 `baud=115200`，无硬件流控，8 位数据，1 位停止位。然后就可以输入“命令表.txt”文件里的命令，可以看到输出的。比如输入命令 `ListRSC semaphore`，超级终端输出如图 1 所示。



```
>ListRSC semaphore
-semaphore for resouce queue
-固定块分配池
-锁控制块池
-wdt pool
-watch dog
-泛设备控制块池
-泛设备句柄池
-read_key
-打开的目录项内存池
-uart0 buf left
-uart0 buf right
-uart0 device left
-uart0 device right
-uart1 buf left
-uart1 buf right
-uart1 device left
-uart1 device right
-samsung k9f2808,3.3v,16Mbytes
-samsung k9f2808,3.3v,16Mbytes

执行成功
>
```

图 1

3 调试

前面介绍的方法你只能看到运行结果，如果你想跟踪 djyos 启动、加载、初始化、运行的全过程，就需要用硬件调试器了。

如果你对嵌入式硬件调试不是很熟悉，请先阅读文档：《建立windows下djyos for arm的编译和调试环境.doc》该文可到www.djyos.com下载。

开始调试之前，先确保 `boot_rom.bin` 文件已经烧录到目标板 0 地址的 flash 中。

这里介绍用 `insight` 和 `rvds` 两种调试环境，这两种环境都支持指令集模拟器，但由于没

有模拟定时器，不能提供 djyos 运行所需要的时钟脉冲，所以不能用指令集模拟调试。

如果你的 2410 或 2440 开发板有选择 norflash 或 nandflash 启动的跳线，务必把跳线跳到 norflash 一侧。

3.1 用 insight 调试

假设你已经按：


《建立 windows 下 djyos for arm 的编译和调试环境.doc》

的介绍安装好了调试环境，并且把 boot_rom.bin 烧录到 0 地址的 flash 中，用 insight 调试的步骤：

- step1: 连接仿真器和开发板，上电运行。
- step2: 执行 JLinkGDBServer.exe 启动 GDB server
- step3: 运行 insight。
- step4: 选取菜单项 file-target settings，按《建立 windows 下 djyos for arm 的编译和调试环境.doc》所述设置。
- step5: 选取菜单项 file-open，打开待调试的文件 debug.elf。
- step6: 选取菜单项 run-connect to target。
- step7: 选取菜单项 run-download。
- step8: 开始调试。

3.2 用 RVDS 调试

RVDS 是 realview2.2 携带的调试器，目前广泛使用的 ads1.2，由于其不能识别 gcc 编译的 elf 格式的执行文件，不能用。realview2.2 是有版权的软件，本文只是介绍如何使用 RVDS 调试，版权问题请自行负责。

假设你已经安装好了 rvds2.2 调试环境，点击图标执行RVDS调试器，按《建立windows 下djyos for arm的编译和调试环境.doc》一文所述连接jlink调试器后，将出现如图 2所示界面。

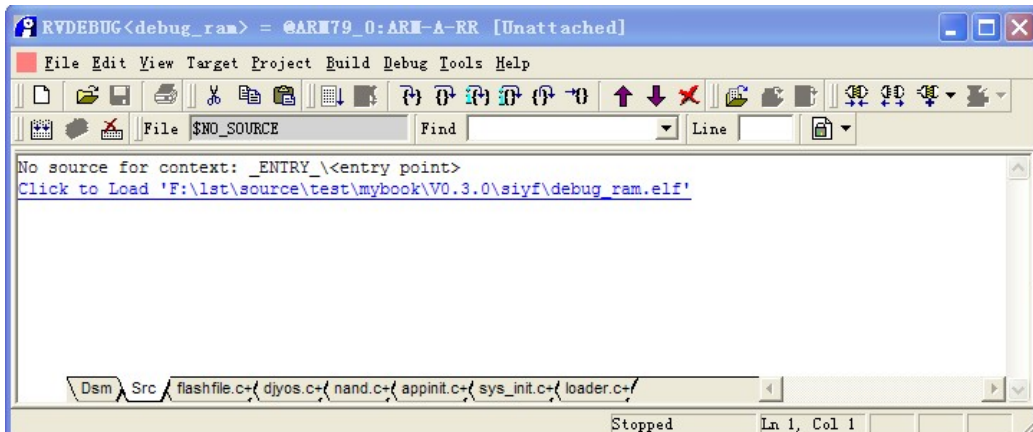


图 2

此时，别忙着加载程序，可能是由于rvds和jlink的匹配问题，直接加载是不能成功的，你需要点击View-Registers菜单项打开registers窗口，如图 3所示，

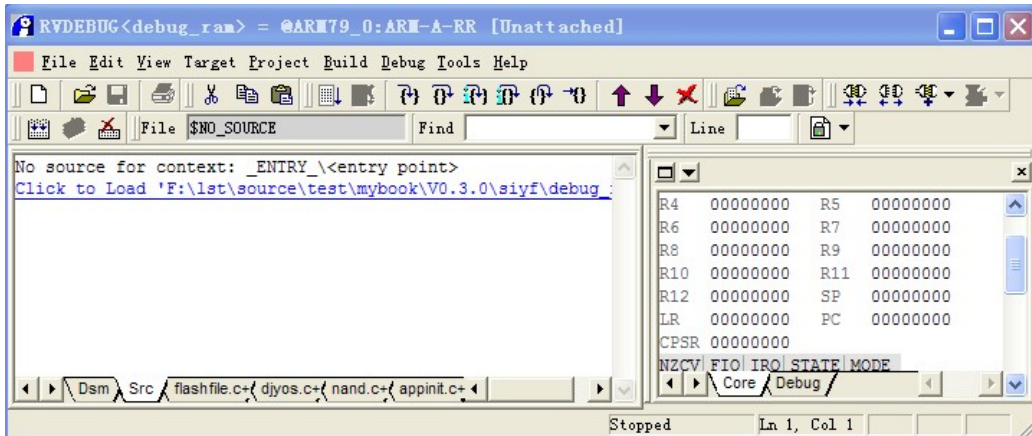
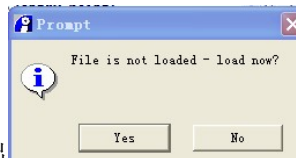


图 3

在图 3所示界面中，

step1、 双击寄存器窗口中 PC 的值，把他改为 0。

step2、 点击 左边的运行按钮，直到右边的按钮亮起，变成这样 ，一次不



行的话，多点击两次。如果弹出 对话框，选择 No 即可。

step3、 点击右边的按钮使 CPU 停下来。

step4、 点击 按钮加载程序映像，从弹出窗口中选择待调试工程的 debug.elf 文件。

调试的过程中，任何时候重新加载代码，均必须从这一步开始。

step5、 重复一遍 1~3 步。

step6、 点击 中最右边的按钮，此时图 3界面将变成图 4所示

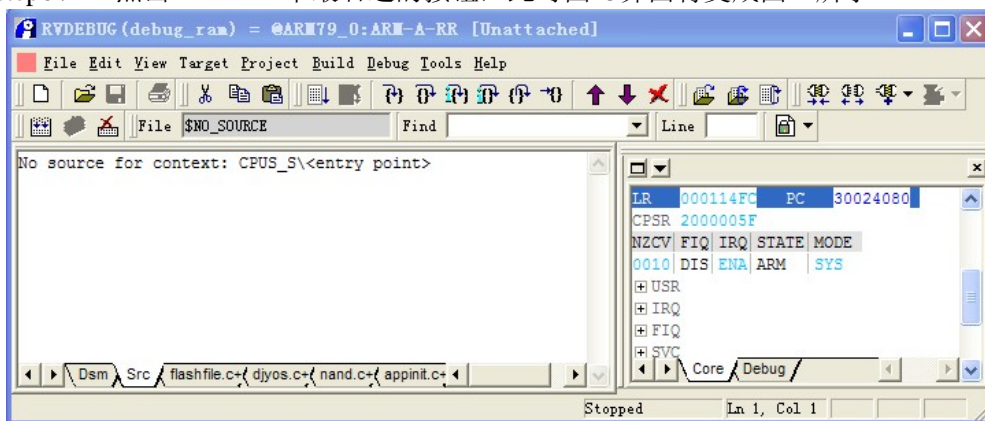


图 4

此时，可以开始调试了，源程序窗口中显示“No source for context:CPUS_S\<entry point>”，是因为程序的入口点在汇编源文件中，直接点 ，将进入源程序调试界面，如图 5所示，图中乱码是因为rvds不能识别汉字注释，不影响调试。

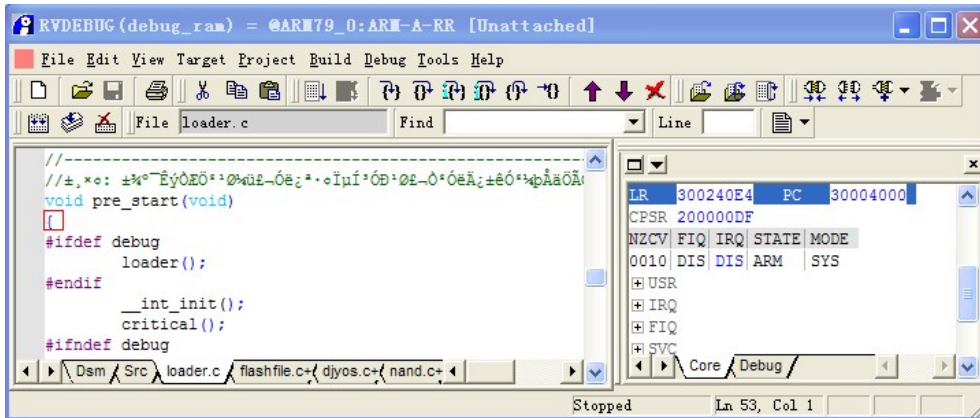



图 5

在调试过程中，可能会发生找不到源程序的问题，弹出如图 6窗口，估计是rvds识别gcc的elf文件的问题，你只要点击图中  图标，人工选择文件就行了，如果老师弹出，则一般关闭rvds窗口，重新运行即可。

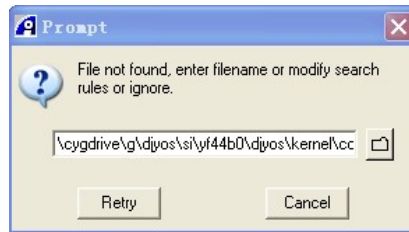



图 6

运行过程中，如果需要重新开始程序，点击  按钮即可，只有在怀疑调试过程中因错误修改了内存中的代码才需要点击中间或左边的按钮，点这两个按钮的话，需要按 step4 开始的步骤加载。