51EDA QT44B0X 开发板说明书

第一部分 基础知识

第一部分 基础知识

第一章 QT44B0X-I 开发套件的组成

1．开发套件所提供的硬件详细清单

QT44B0X-I 开发套件包括：

1) 一块已测试好的QT44B0X-I 开发板；

2）一个3合1的JTAG调试头；

3）一条串口线(两边都是母头，2与3交叉)；

4）一条并口线（一边是公头，一边是母头，一对一）；

5）一条USB线(一边是A型，一边是B型)；

6）一条网线(交叉线)；

7) 一张QT44B0X-I光盘；

8) 一个+12V直流电源；

9）一个包装盒；

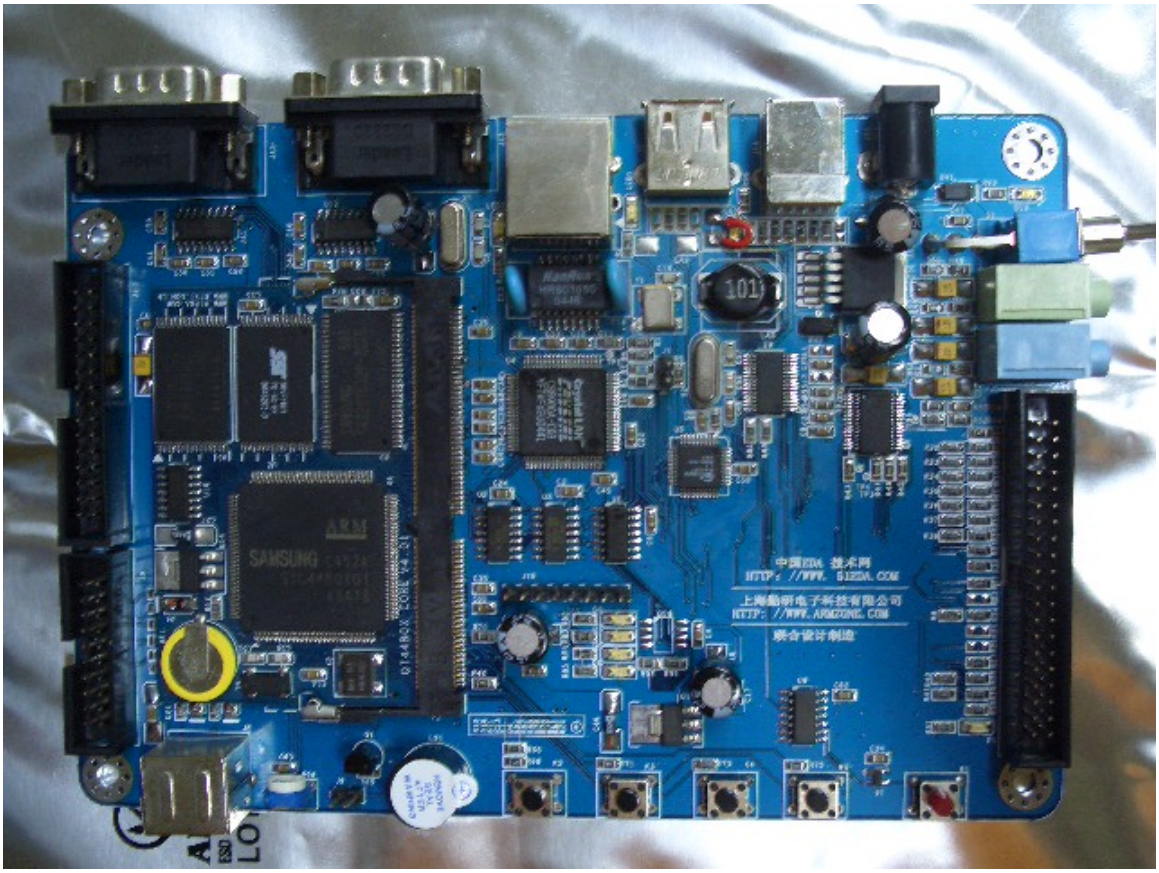

SAMSUNG
ARM
中国EDA 技术网
HTTP: //WWW. 51EDA. COM
HTTP: //WWW. ARMZONE. COM

图 1-1

2. 开发套件所提供的软件详细清单

1) ADS1.20安装程序(评估版)；

2) 使用3合1的JTAG调试头并支持ADS1.20 的JTAG 调试软件CRDI；

3) 烧写Flash的工具软件flashpgm2.2.4（评估版），以及适用于S3C44B0的OCD 配置文件；

4) 烧写Flash的工具软件FlashP（正式版），此软件版权属51EDA；

5) 串口工具软件sscom32.exe和网络上传下载软件tftp.exe；

6) LCD 图片转换和字模转换工具软件uC-GUI-BitmapConvert.exe；

7) QT44B0X-I BIOS源代码（ADS1.20 的项目文件，包含CS8900A驱动和TFTP协议源码）；

8) QT44B0X-I测试程序(ADS1.20 的项目文件，包含全部源代码)，具有下列功能测

试：

- 内存（SDRAM）读写自测试；
- PWM输出蜂鸣器测试；
- IIC总线EEPROM读写测试；
- 模数转换ADC测试；
- USB从设备PDIUSBD12测试；
- 黑白STN液晶屏测试；
- 4 级灰度STN液晶屏测试；
- 16 级灰度液晶屏测试；
- 256色液晶屏测试；
- STN屏字符串显示测试；
- 外部中断测试；
- PS/2接口测试；

9）三星提供的S3C44B0标准测试程序，经修改后可以在QT44B0X-I开发板上运行；

****

★SAMSUNG S3C44B0X开发板
★中国EDA技术网 www.51EDA.com

S3C44B0X测试程序 V1.0
系统主频 = 64MHz
串口选择COM0
串口波特率 = 57600bps
本项目编译时间：Jun 19 2005-14:59:34

Image$$RO$$Base = c000000
Image$$RO$$Limit = c01313c
Image$$RW$$Base = c01313c
Image$$RW$$Limit = c01393c
Image$$ZI$$Base = c0132c8
Image$$ZI$$Limit = c01393c

****

★中国EDA技术网 www.51EDA.com S3C44B0X开发板测试程序:

0:About	1:Lcd Mono	2:Lcd G4	3:Lcd G16
4:SL_IDLE Mode	5:SL_IDLE Mode20	6:IDLE Mode	7:IDLE(hard)
8:nWAIT pin	9:nXDREQ0	10:Cache	11:Adc 0,1,2,3
12:Adc with DMA	13:UART 0	14:UART 0 FIFO	15:UART 1
16:UART 1 FIFO	17:SLOW Mode	18:HOLD Mode	19:STOP Mode
20:Zdma0	21:Zdma1	22:SIO Tx/Rx	23:SIO Tx BDMA0
24:SIO Rx BDMA1	25:WDTimer	26:RTC(display)	27:RTC(Test)
28:RTC Tick	29:IIC(KS24C080)	30:IIS(uda1341)	31:IIS Tx(Slave)
32:IIS Rx(Master)	33:Timer Int	34:Tout test	35:Ext. Int
36:Etc...	37:Change PLL	38:Test AFC(Tx)	39:Test AFC(Rx)
40:Write flash			

请输入数字选择一项功能进行测试:

图 1-2

10）uCLinux 源码包，包含CS8900A网卡驱动、SL811HST驱动、IDE硬盘驱动、NandFlash驱动、串口驱动、可读写的NandFlash文件系统，支持格式为yaffs和jffs2；

11）使用3合1的JTAG进行硬件仿真DEBUG的演示动画；

12）使用FLASHPGM烧写Flash的演示动画；

13）使用FLASHP烧写Flash的演示动画；

14) QT44B0X-I原理图（核心板和底板图纸，SCH和PDF格式）；

15) QT44B0X-I开发板使用手册

3. 板上接口和资源清单

- 中央处理器
  - S3C44B0X (Samsung)，ARM7TDMI
- 外部存储器
  - 2M Bytes Nor Flash
  - 16M Bytes SDRAM
  - 16M Bytes Nand Flash
  - 10M 网口，CS8900A-CQ3
- LCD 接口
  - 320x240,STN ,16 级灰度, 最大可接 640x480 256 色
- USB Device 接口
  - USB1.1 规范，PDIUSBD12
- USB Host 接口
  - USB1.1 规范，SL811HST
- 串口
  - 两个标准 9 针 RS232 接口
- 时钟源
  - 内部实时时钟（备有掉电电池）
- IDE 接口
- 音频输出
  - UDA1341
- 四个小按键，四个 LED
- 一个蜂鸣器
- 一个 PS/2 接口

4．板上硬件资源分配列表

1）系统片选及地址空间

nGCS0【0x0000_0000】：FLASH（SST39VF1601）

nGCS1【0x0200_0000】：NANDLASH（K9F2808）

nGCS2【0x0400_0000】：PDIUSBD12（USB DEVICE）

nGCS3【0x0600_0000】：CS8900A

nGCS4【0x0800_0000】：SL811HST（USB HOST 或 USB DEVICE）

nGCS5【0x0A00_0000】： IDE/ATA

nGCS6【0x0C00_0000】：SDRAM（K4S641632F 或 HY57V641620）

2）中断分配

EINT0：PDIUSBD12（USB DEVICE）

EINT1：SL811HST（USB HOST 或 USB DEVICE）

EINT2：IDE/ATA

EINT3：CS8900A

EINT4：KEY2

EINT5：KEY3

EINT6：KEY4(与PS/2接口共用)

EINT7：KEY5（与PS/2接口共用）

3）系统板设定：

BANK0 总线宽度为16 bit；

Little Endian 模式；

跳线J9 禁止/使能蜂鸣器（插上短路块蜂鸣器鸣叫使能）；

4）板上接口和指示灯功能说明

- 接口板

D1：IDE/ATA接口访问指示灯；

D4：USB Device指示灯；

D6：由IO 口GPE7控制的指示灯；

D7：由IO 口GPE6控制的指示灯；

D8：由IO 口GPE5控制的指示灯；

D9：由IO 口GPE4控制的指示灯；

D10：12V 直流电源指示灯；

LED0：10M 网口RX 发送指示灯；

LED1：10M 网口Link 联机指示灯。

J1：网卡芯片的SLEEP模式短路块，正常状态下不接；

J2：USB－A型接口，USB HOST接口；

J3：40PIN插座，IDE/ATA硬盘电缆接口；

J4：IIS 立体声输出音频插座；

J5：IIS音频输入(麦克风)插座；

J6：20PIN插座，标准JTAG电缆接口；

J7：30PIN插座，LCD和触摸屏电缆接口；

J8：PS/2接口，接PS/2键盘或鼠标；

J9：2PIN短路块，禁止/使能蜂鸣器（插上短路块蜂鸣器才会鸣叫）；

J11：12V直流电源输入插座；

J12：DB9，COM0接口

J13：DB9，COM1接口

J14：USB－B型接口，USB DEVICE接口；

J15：8PIN插座，模数转换信号引脚输入接口；

P1：网卡接口。

第二章 板上资源测试方法和步聚

1、超级终端的设置

在 Windows 桌面左下角依次点击：开始----程序----附件----通讯----超级终端，第一次运行此程序系统会提示下面窗口：

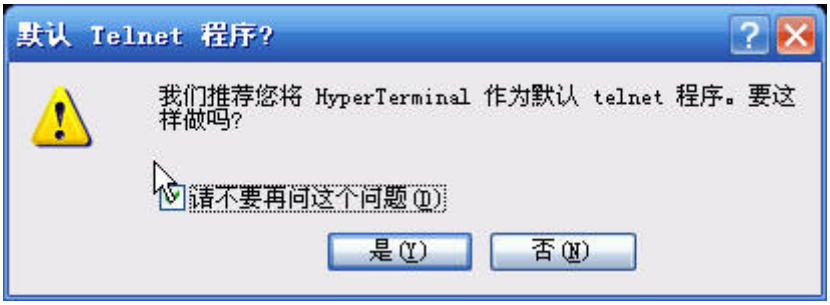


图 2-1

为了以后方面，在“请不要问这个问题”选项中选中，以后打开的时候就不会出现这样的提示，再点击“是”。会出现一些基本区号等信息设置，随便填写就可以了。超级终端会提示一个对话框，任意输入一个名称与选择一个图标，点“确定”。

图 2-2

接下来就是设置你的串口，系统会列出你的所有串口，你选择一个没有被使用的串口，一般的用户都是串口 1。

图 2-3

再进行一些设置，这些设置必须按照下面的要求来设置，否则是没有办法工作：

- 每秒位数：57600
- 数据位：8
- 奇偶校验：无
- 停止位：1
- 数据流控制：无

图 2-4

超级终端配置完成，开发板上电后，可以在超级终端中看到如下输出信息：

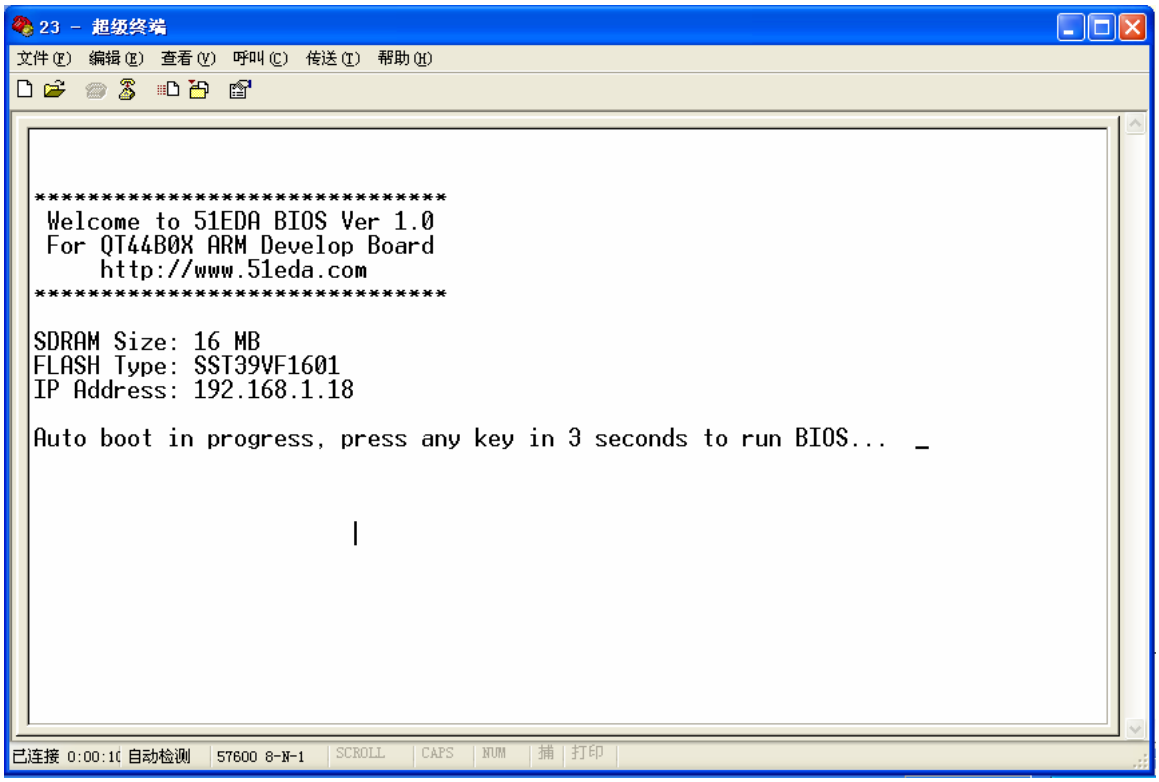


图 2-5

2 、如何运行测试程序

将配套光盘中“烧写文件\测试文件”文件夹下的 51EDAQT44B0_TEST.elf 文件用 FlashPgm 软件烧到开发板上，具体步骤请参考第四章的内容。

程序烧写到开发板后，接上网线和串口电缆，按上一节的步骤设置超级终端，开发板加电后就会运行测试程序，下图是测试程序的主菜单：

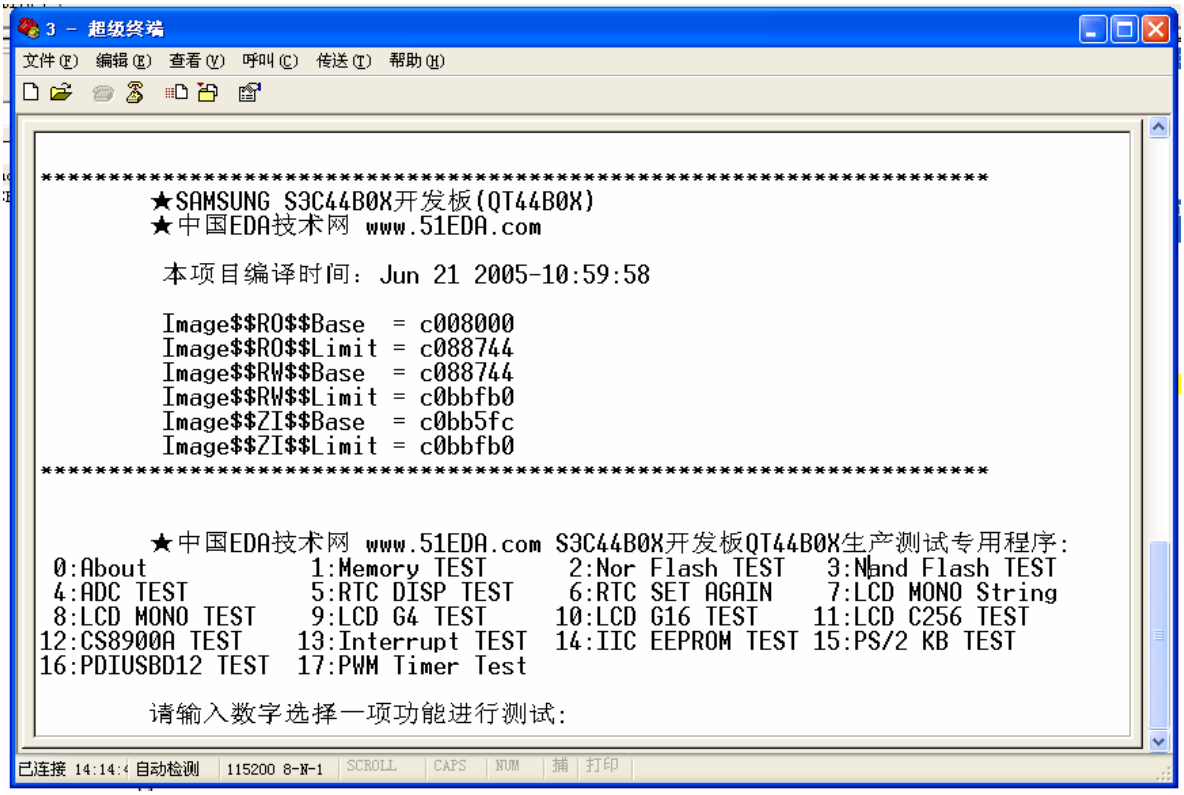


图 2-6

3、内存 SDRAM 读写测试

按 1 然后回车进行内存（SDRAM）自读写测试，提示信息如下：

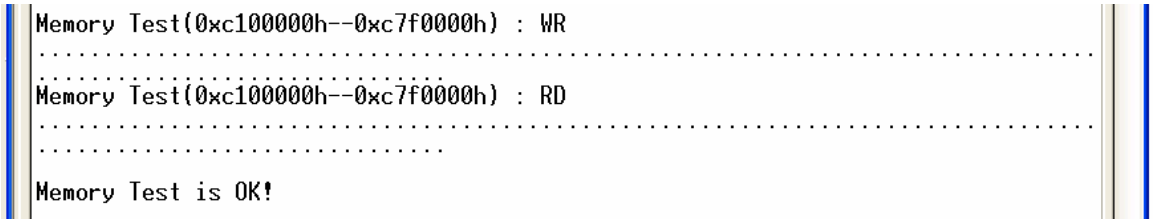


图 2-7

4、Nor Flash 测试

按 2 然后回车后，测试程序会读出 FLASH 的 ID 号，提示信息如下：

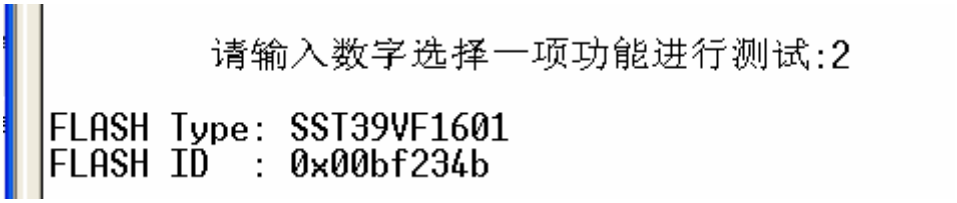


图 2-8

5、Nand Flash 测试

按 3 然后回车后，测试程序会读出 Nand Flash 的 ID 号，提示信息如下：

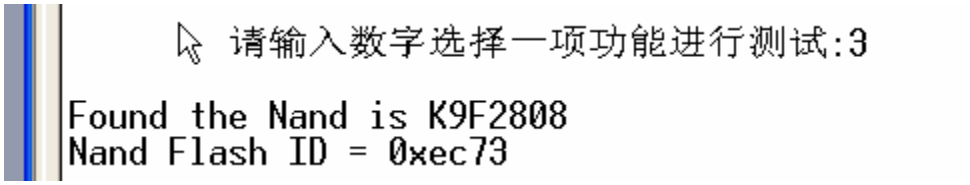


图 2-9

6、AD 转换测试

按 4 然后回车后，进行 AD 转换的测试，开发板的 J15 脚引出了 8 个 AD 输入脚，改变其上的电压值可以看到输出数值发生变化，如下图所示：

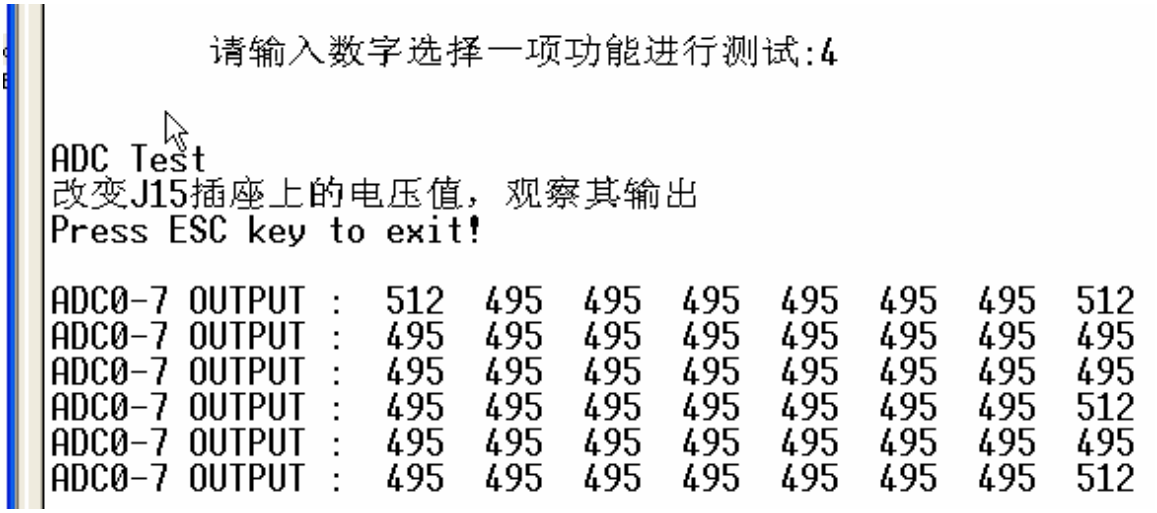


图 2-10

7、RTC（实时时钟）测试

按 5 然后回车后，将显示开发板记录的日期与时间，显示结果如下图所示：

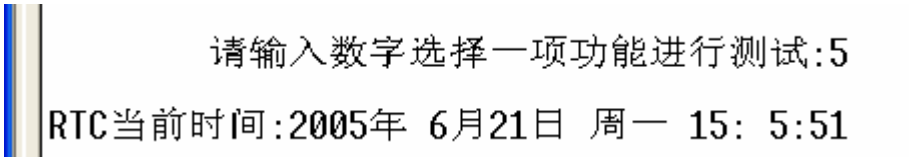


图 2-11

8、RTC（实时时钟）设置测试

按 6 然后回车后，可以设置开发板的日期与时间，操作如下图所示：

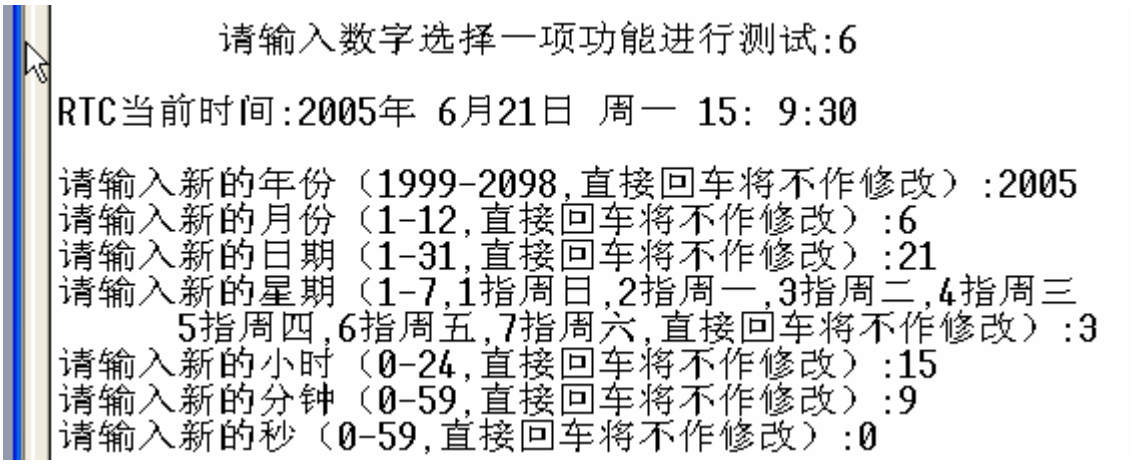
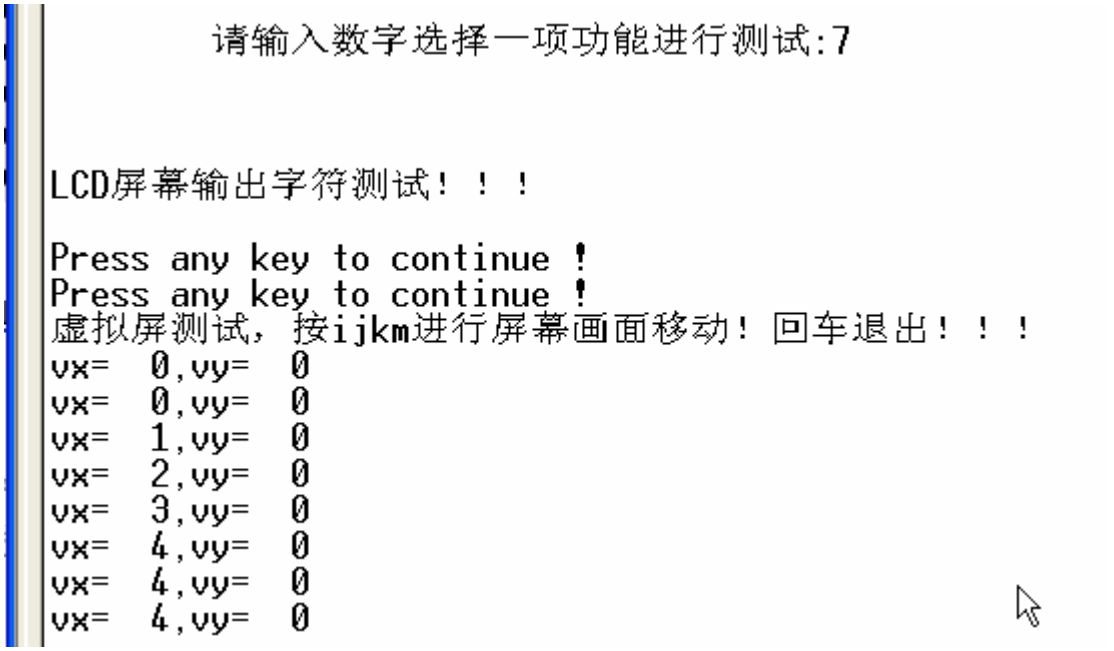


图 2-12

9、黑白 STN 液晶屏显示英文字符测试

按 7 然后回车后，可以进行黑白 STN 液晶屏的字符串测试，测试输出如下：


请输入数字选择一项功能进行测试:7
LCD屏幕输出字符测试！！！
Press any key to continue !
Press any key to continue !
虚拟屏测试，按ijkm进行屏幕画面移动！回车退出！！！
vx= 0,vy= 0
vx= 0,vy= 0
vx= 1,vy= 0
vx= 2,vy= 0
vx= 3,vy= 0
vx= 4,vy= 0
vx= 4,vy= 0
vx= 4,vy= 0


www.51eda.com www.armzone.com

图 2-13

10、黑白 STN 液晶屏测试

按 8 回车后，开始测试黑白的 STN 液晶屏，接上黑白 STN 液晶屏，按照提示信息按回车或者是“i”、“j”、“k”，“m”，可以看到液晶屏上面出现不同的画面，相关提示信息如下：

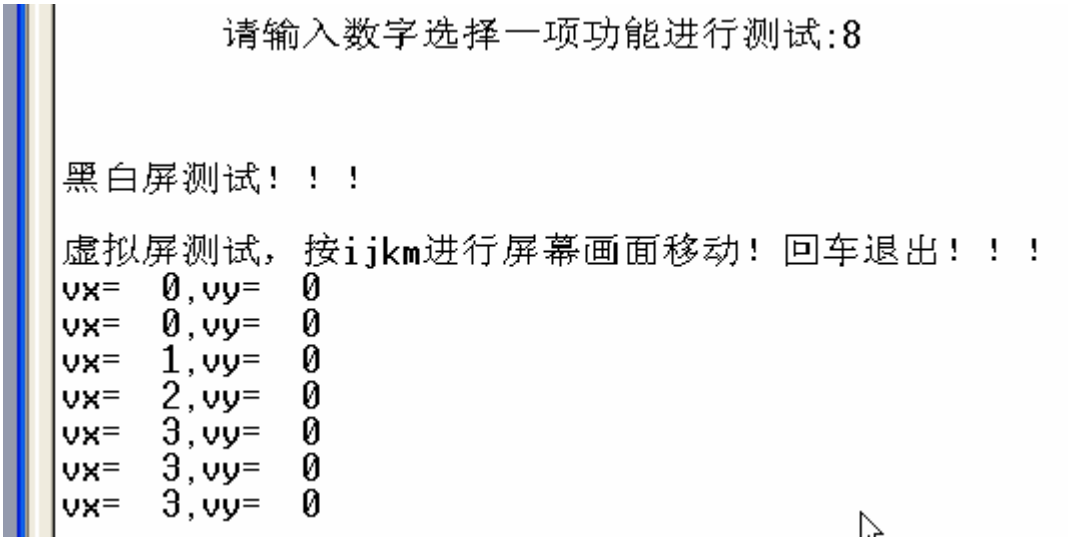


图 2-14

11、四色液晶屏测试：

按 9 回车后，开始测试四色液晶屏，接上四色液晶屏，按照提示信息按回车或者是“i”、“j”、“k”，“m”，可以看到液晶屏上面出现不同的画面，相关提示信息如下：

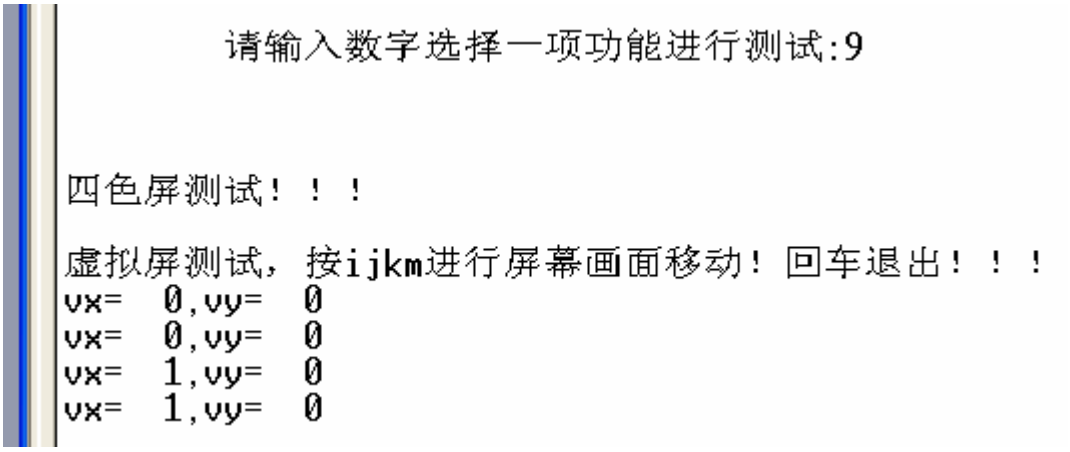


图 2-15

12、四色液晶屏测试：

按 10 回车后，开始测试十六色液晶屏，接上液晶屏，按照提示信息按回车或者是“i”、“j”、“k”，“m”，可以看到液晶屏上面出现不同的画面，相关提示信息如下：

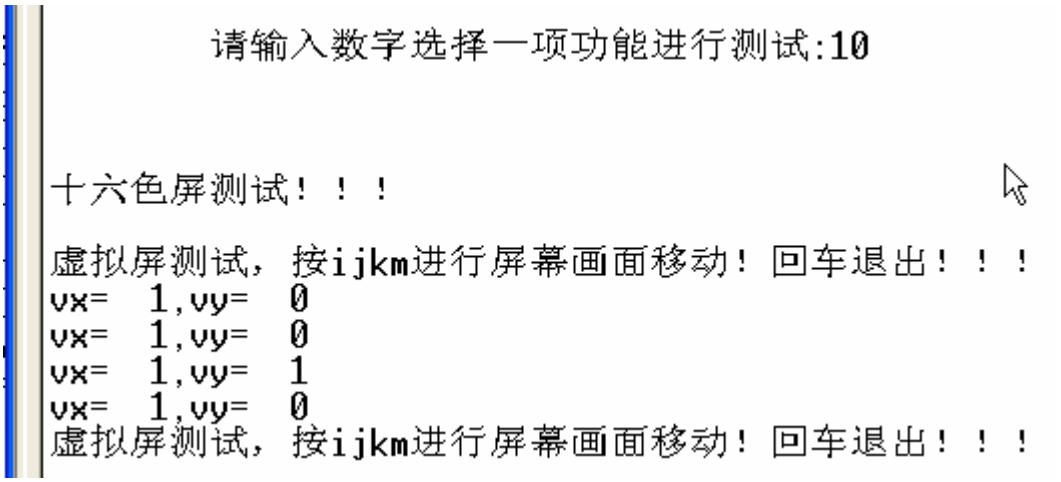
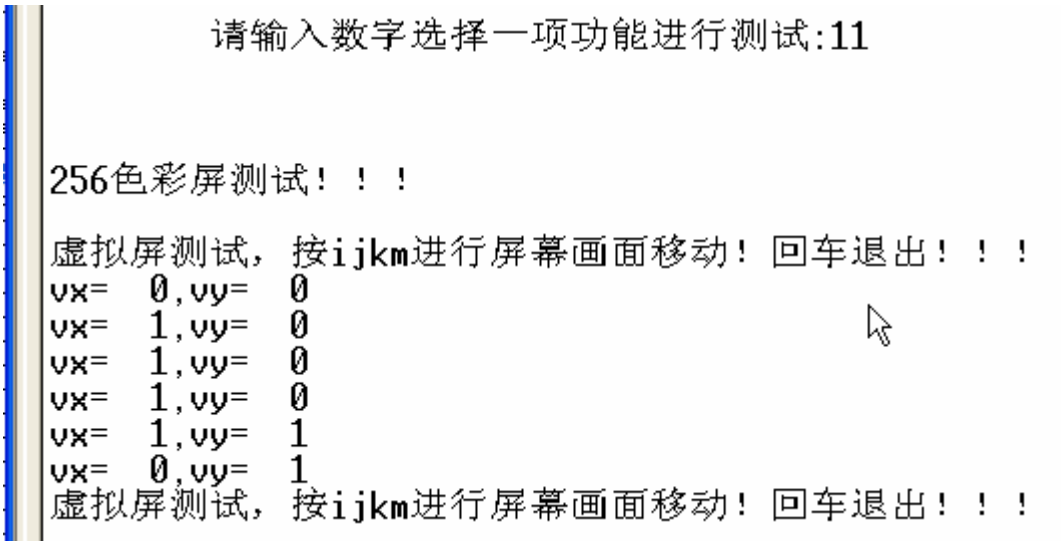


图 2-16

13、256 色液晶屏测试：

按 10 回车后，开始测试 256 色液晶屏，接上液晶屏，按照提示信息按回车或者是“i”、“j”、“k”，“m”，可以看到液晶屏上面出现不同的画面，相关提示信息如下：



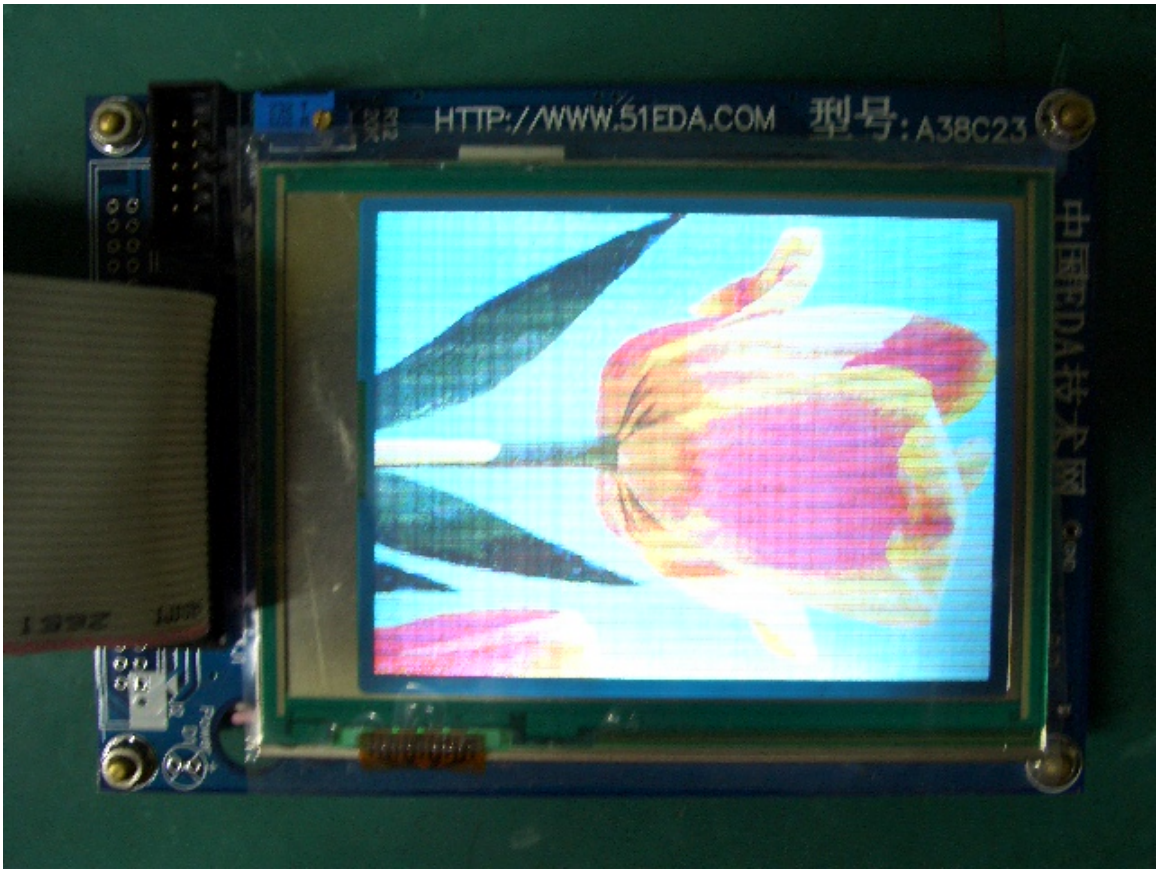


图 2-17

14、网卡测试

在执行这次测试时，请用网线把 PC 和开发板连接上，并把 PC 机的 IP 设为 192.168.1.x（x 为 1-255，8 除外的整数，8 已经被开发板所用），子网掩码设为 255.255.255.0。

在超级终端中输入 12 回车后，提示信息如下图所示：

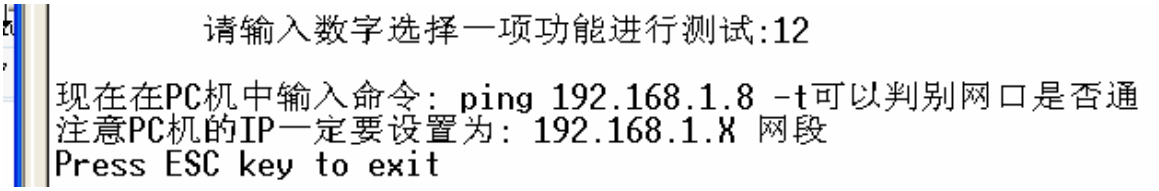


图 2-18

然后在 PC 机上输入指令 ping 192.168.1.8，如果出现如下信息，则说明网卡工作正常。

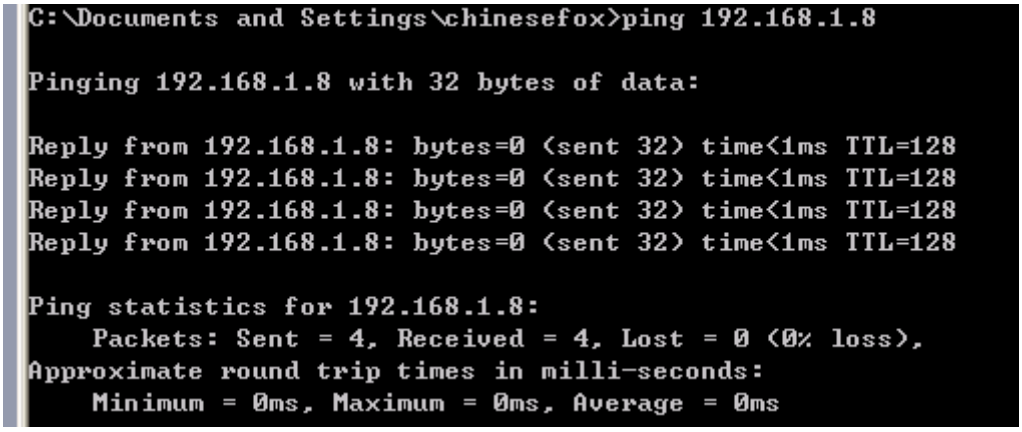


图 2-19

15、外部中断测试

输入 13 回车，程序测试外部电平中断，按板子上面的按键 K2、K3 就可以看到中断响应信息，相关提示信息如下（注：由于 K4、K5 与 PS/2 相连，因此不能响应中断请求）：

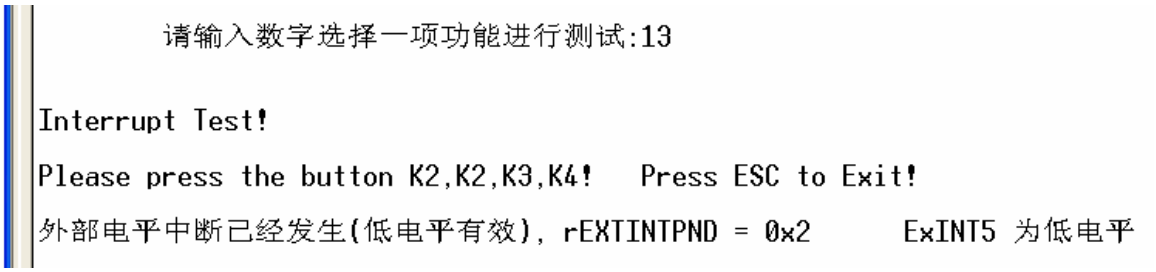


图 2-20

16、IIC 总线 EEPROM 测试

输入14回车，可以测试板上的IIC总线EEPROM，提示信息如下图如示：

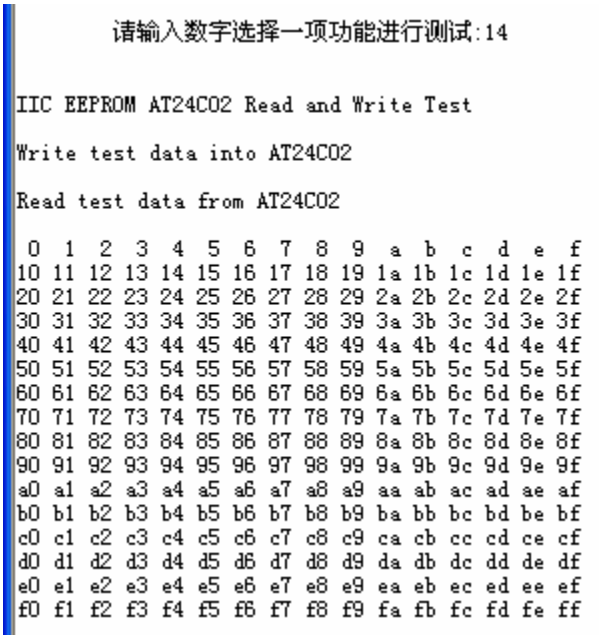


图 2-21

17、PS/2 键盘测试

先把 PS/2 接口的键盘接到开发板上，然后输入 15 回车，程序开始测试 PS/2 键盘，输出信息如下：

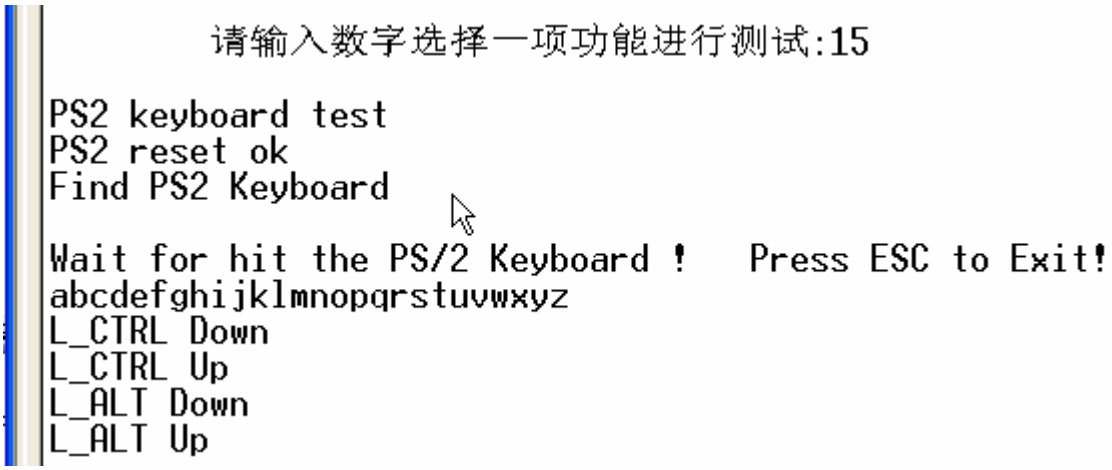


图 2-22

18、USB Device 测试

在主菜单中输入 16 回车后，可以看到如下的提示信息：

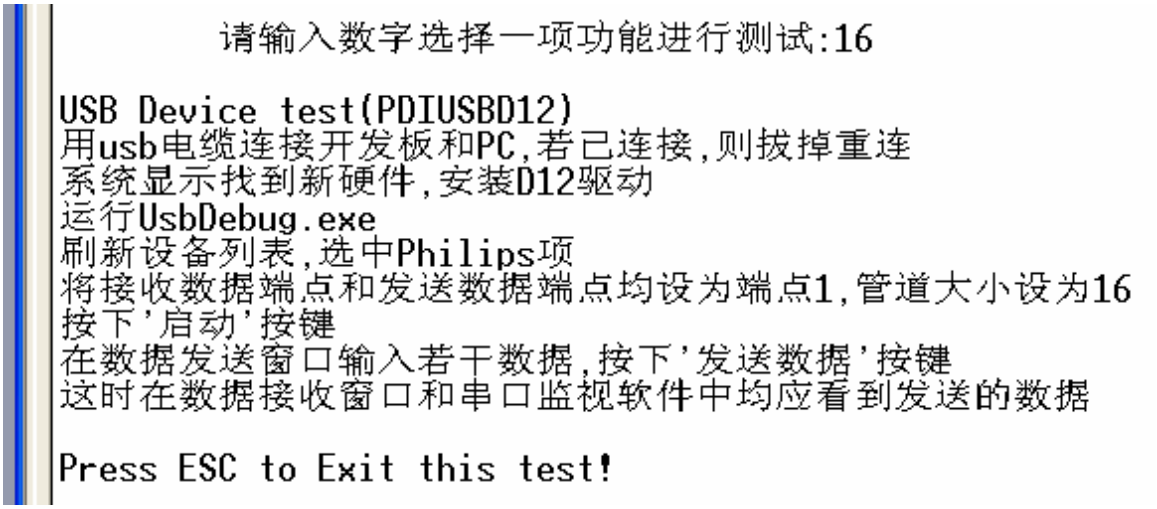


图 2-23

按提示的要求：用 usb 电缆连接开发板和 PC,若已连接，则拔掉重连系统显示找到新硬件，安装 D12 驱动。上述文件都存放在配套光盘里的 Tools\USB 文件夹下。

驱动安装过程如下，当把 USB 线插到 PC 中时，会弹出下面的窗口，选择“否，暂不”，然后点击下一步按钮。

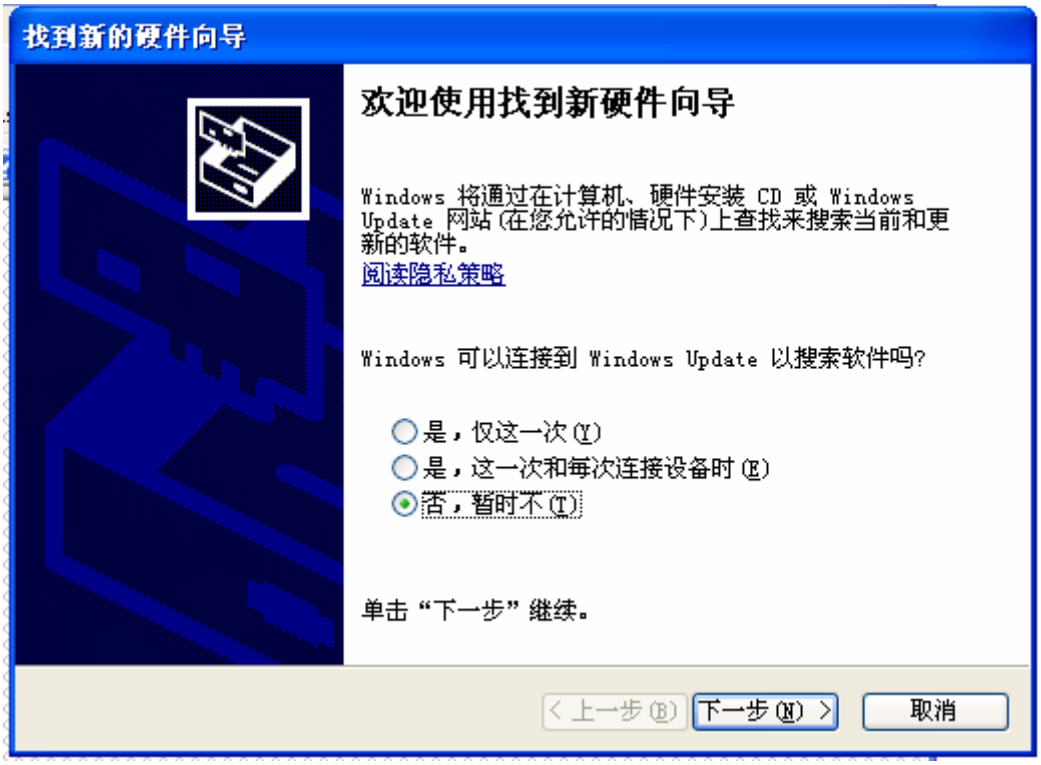


图 2-24

选择“从列表或指定位置安装”，并点击“下一步”按钮。

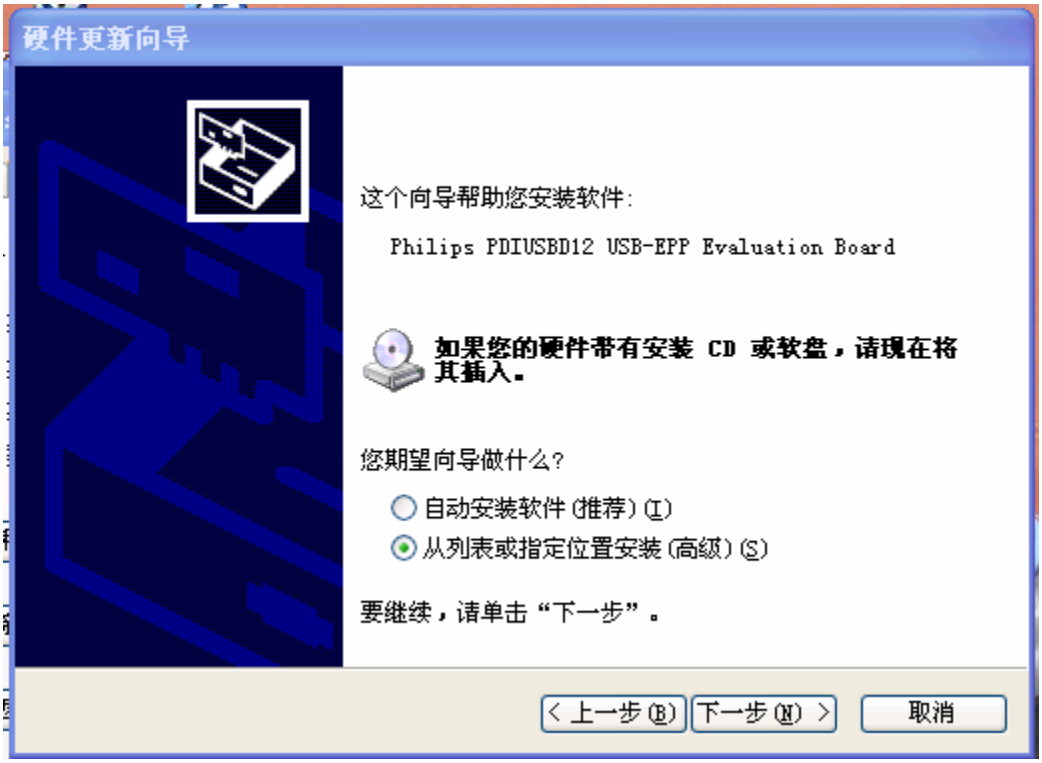


图 2-25

手动定位驱动程序，该驱动程序位于配套光盘中 Tools\USB\usb 驱动\D12 驱动\win2k_xp 文件夹下。

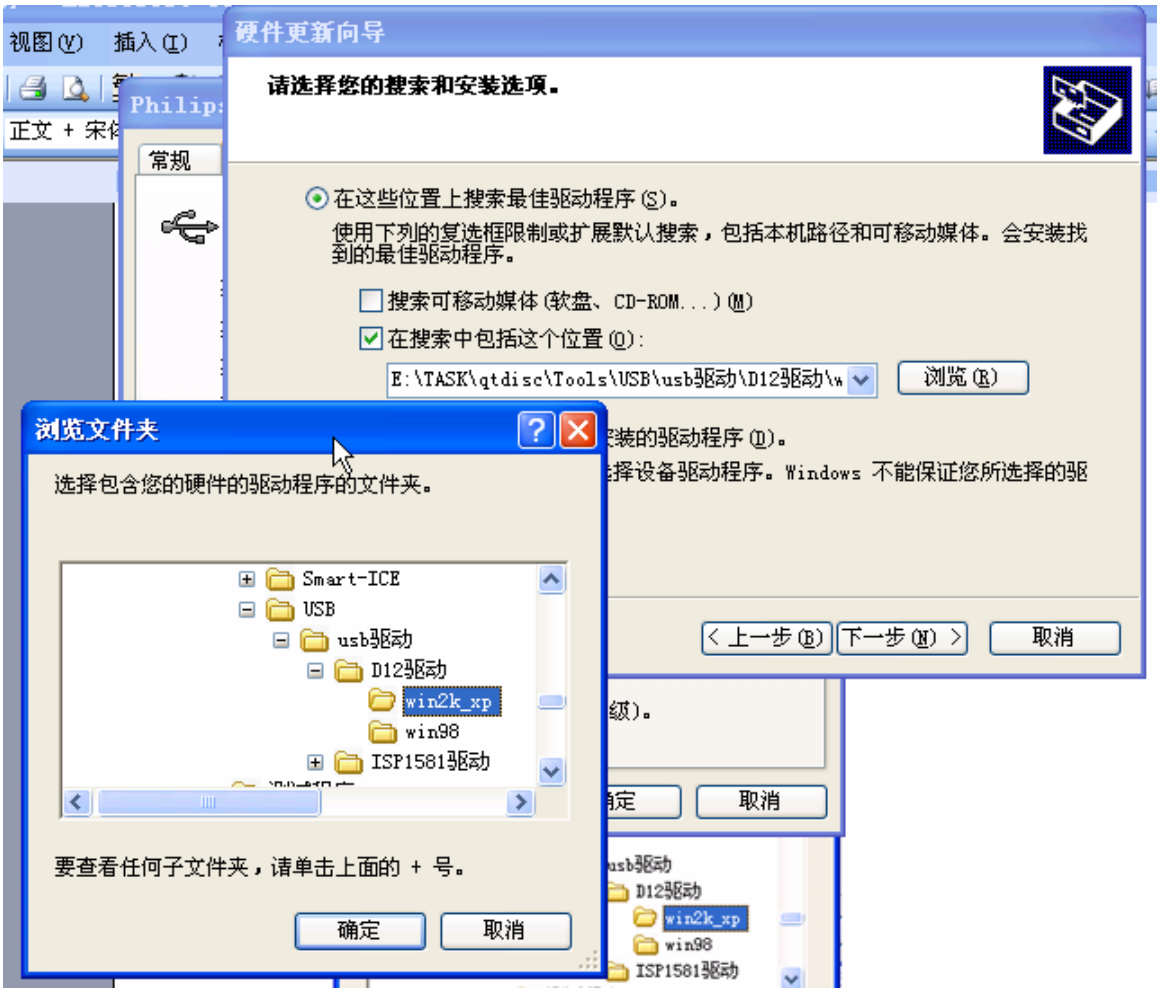


图 2-26

当出现如下界面时，点击”仍然继续“按钮，至此，D12 的驱动程序安装完毕。

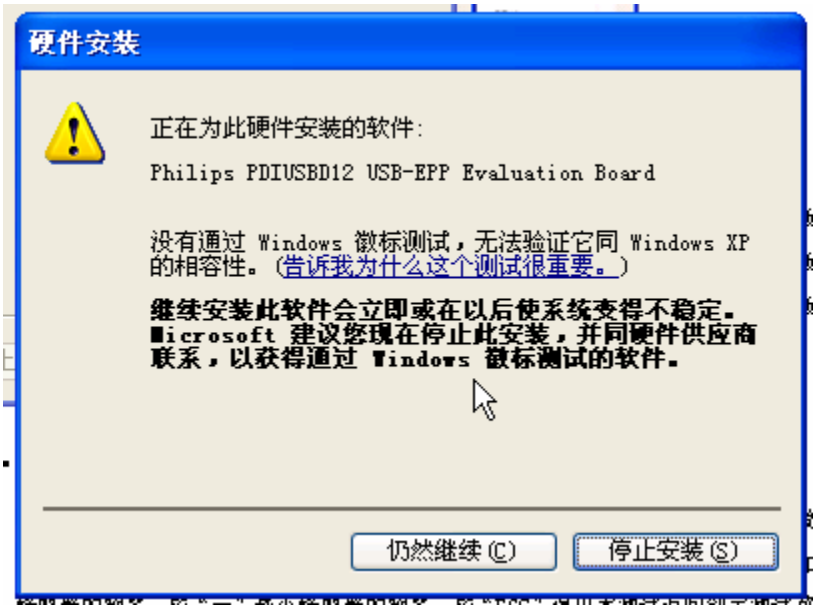


图 2-27

运行 UsbDebug.exe，刷新设备列表，选中 Philips 项将接收数据端点和发送数据端点均设为端点 1,管道大小设为 16 按下’启动’按键在数据发送窗口输入若干数据,按下’发送数据’按键这时在数据接收窗口和串口监视软件中均应看到发送的数据，结果如下图所示:

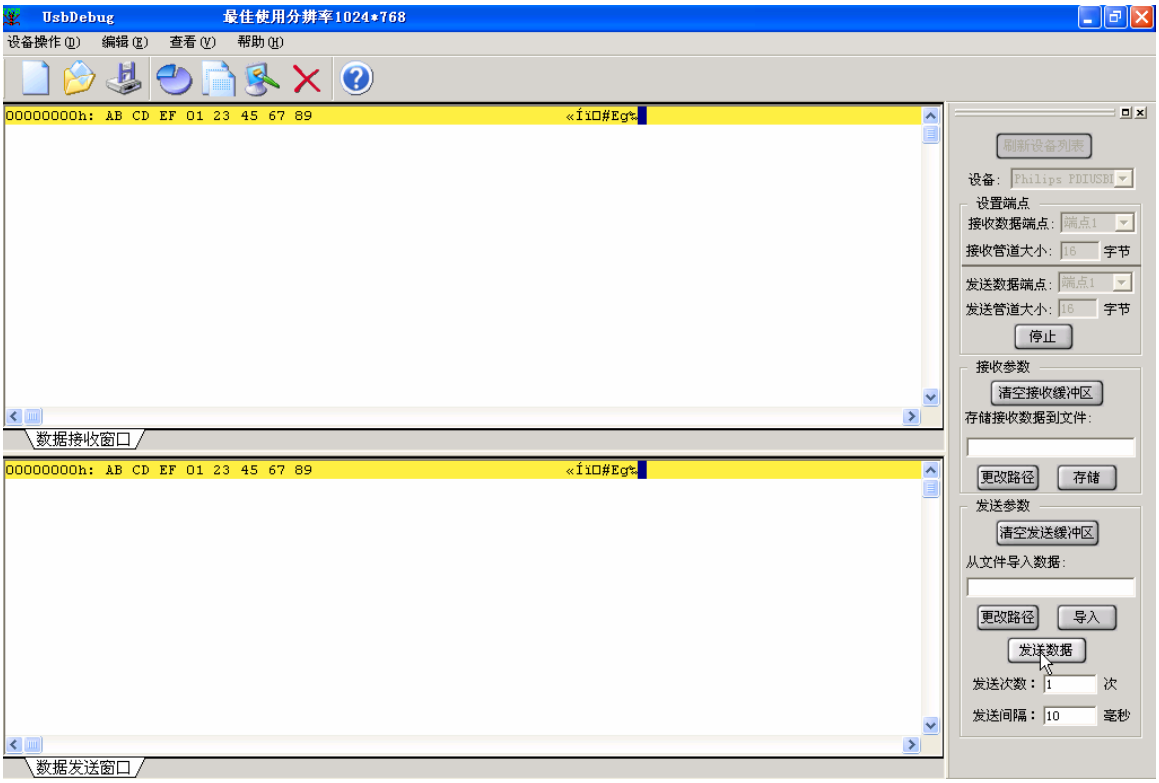


图 2-28

19、PWM 脉宽调试和蜂鸣器测试

在 J6 上接上一个短路块，使能蜂鸣器，然后回车输入 17 回车，测试 PWM（脉宽调制）输出并驱动一个蜂鸣器，现在就可以听到蜂鸣器的鸣叫声了，按“＋”增加蜂鸣器的频率，按“－”减小蜂鸣器的频率，按“ESC”退出本测试返回到主测试菜单，提示信息如下图：

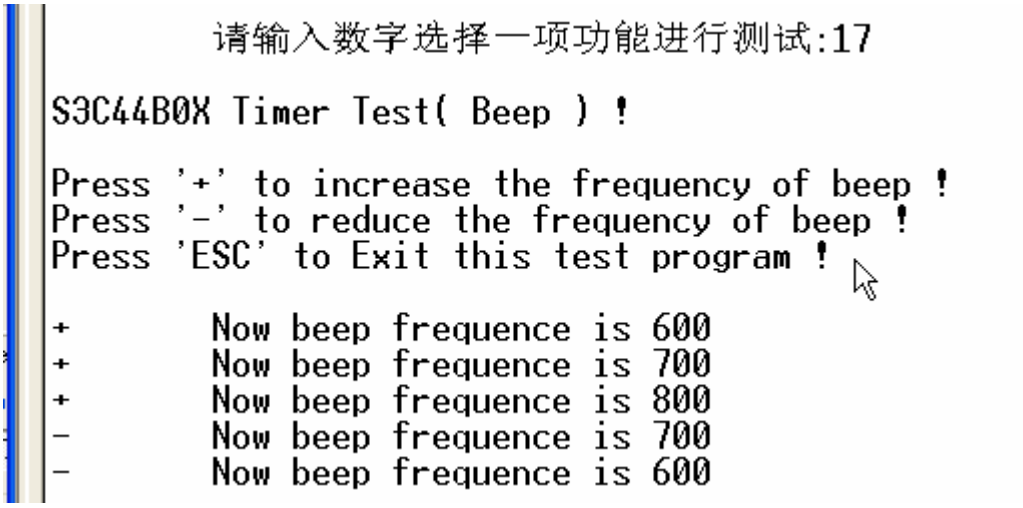

请输入数字选择一项功能进行测试:17
S3C44B0X Timer Test(Beep) !
Press '+' to increase the frequency of beep !
Press '-' to reduce the frequency of beep !
Press 'ESC' to Exit this test program !
+ Now beep frequence is 600
+ Now beep frequence is 700
+ Now beep frequence is 800
- Now beep frequence is 700
- Now beep frequence is 600

图 2-29

第三章 QT44B0X 调试指南

1、安装 ADS1.20 编译调试环境

在配套光盘的 Tools 文件中，我们提供了 ADS1.20，双击 SETUP 图标后进行安装

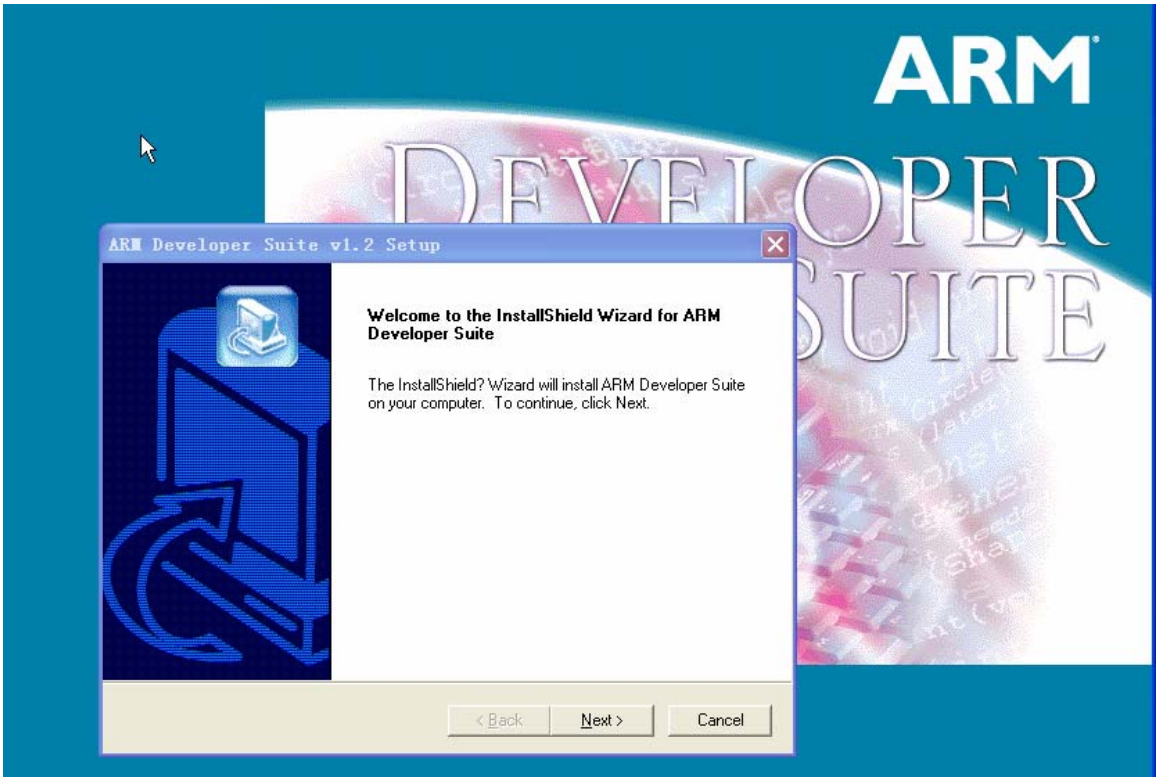


图 3-1

安装结束后，会弹出一个对话框，要求输入 License：

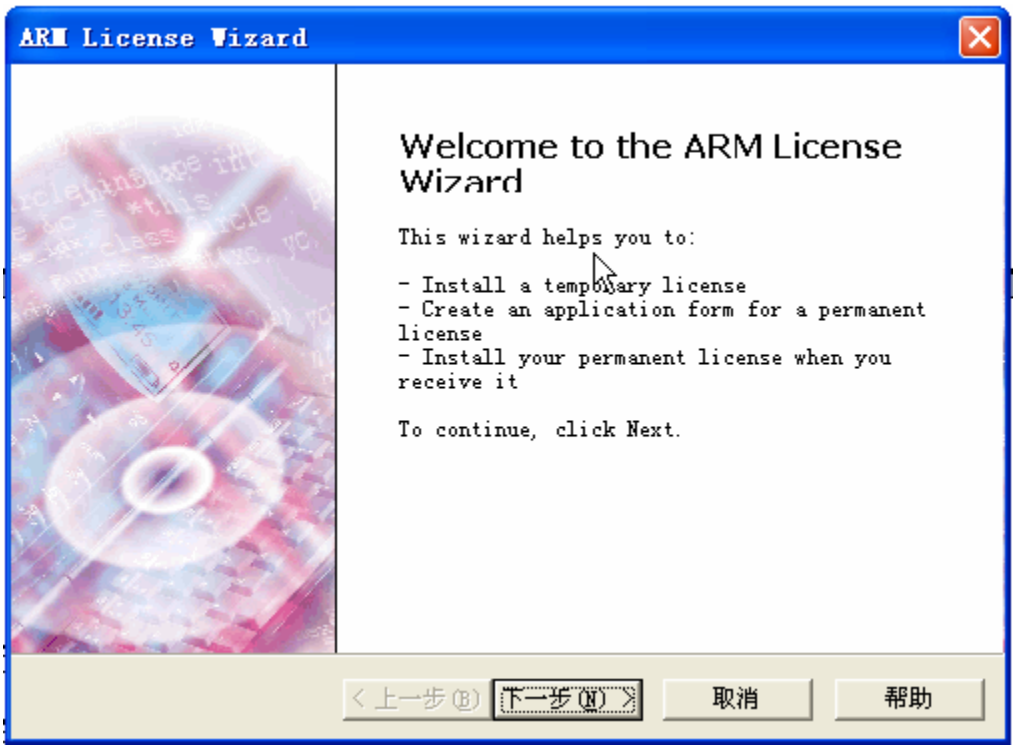


图 3-2

点击两次“下一步”后，出现如下图所示的对话框

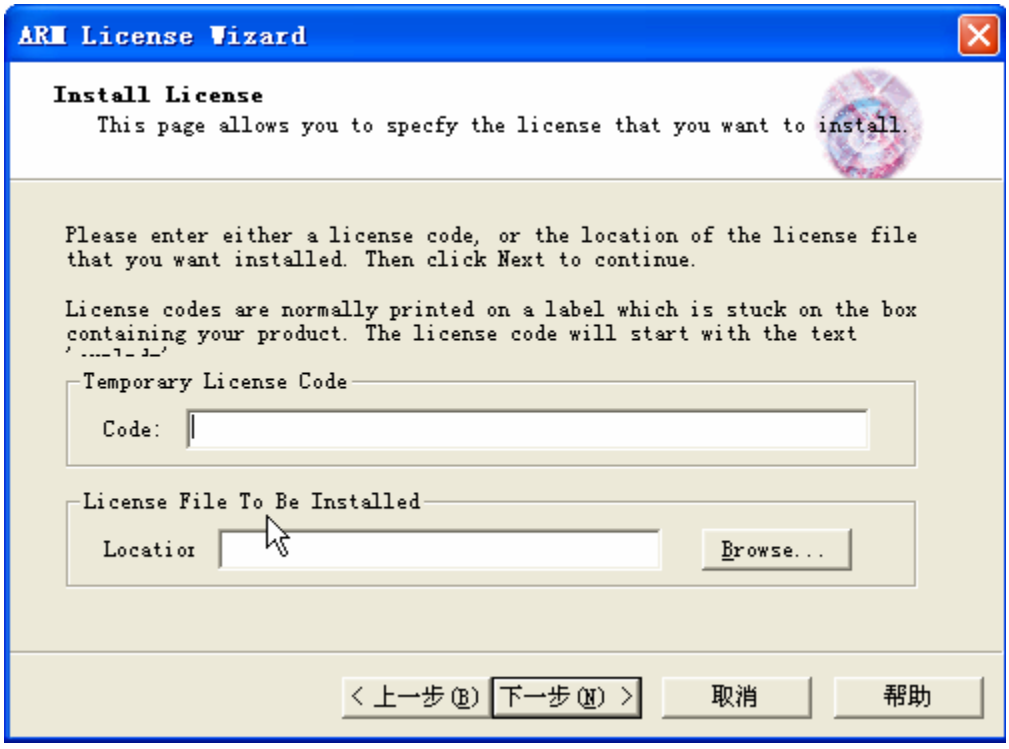


图 3-3

点击 Browse...按钮，并选择 CRACK 文件夹下的 LICENSE 文件

图 3-4

点击“下一步”后，ADS1.20 安装完毕

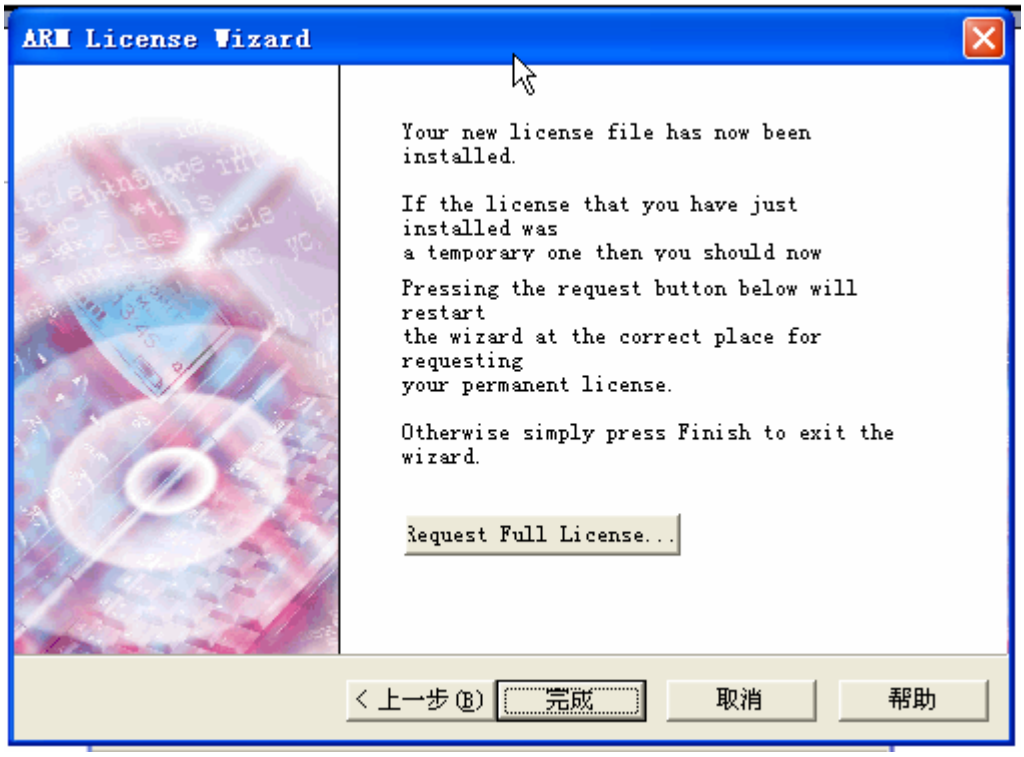


图 3-5

2、使用 3 合 1 的 JTAG 在 ADS1.20 环境下进行仿真调试

1）仿真环境的准备

- 安装简易 JTAG 的 RDI 接口文件

Win2K/XP 用户须安装 giveio 驱动,这是因为 Win2K/XP 保护机制,运行于系统下的程序不能直接访问 IO 口,CRDI.DLL 通过 giveio 驱动来获得 IO 的使用许可。WIN98 则无此限制。

用户需要先安装配套光盘中 tools 文件夹下的 giveioinstaller 程序, 点击 Install Service 即可。然后将光盘所附的 CRDI2.0.RAR 用 WINRAR 解压缩到本机任一目录(如 c:\crdi\)。

图 3-6

- 打开 AXD Debugger 进行设定

启动 AXD Debugger

图 3-7

选择 Options->Configure Target…

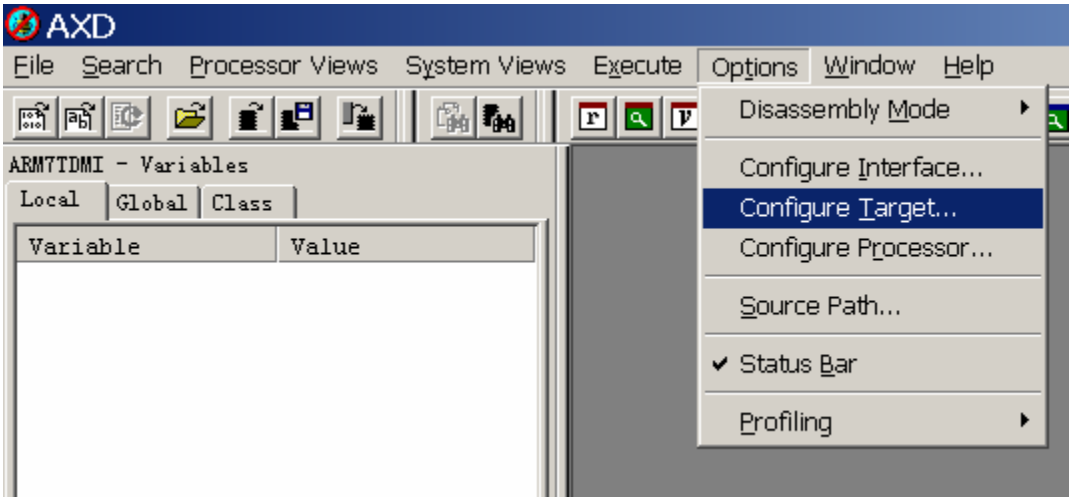


图 3-8

在弹出对话框点击 Add

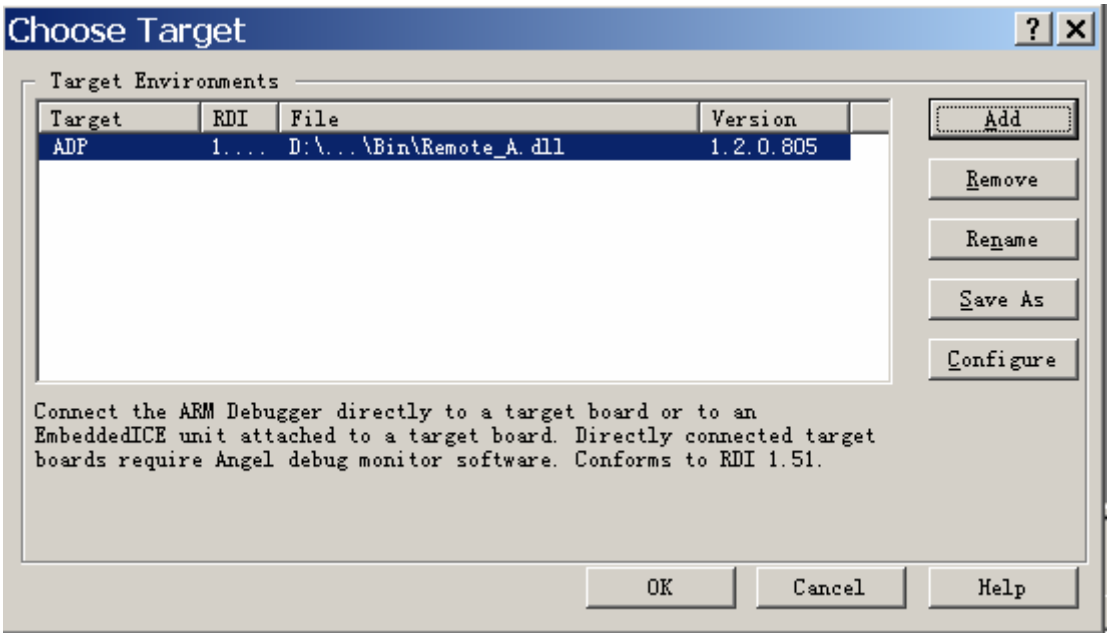


图 3-9

进入 CRDI 解压缩路径，选择 CRDI.dll 文件，点击打开。

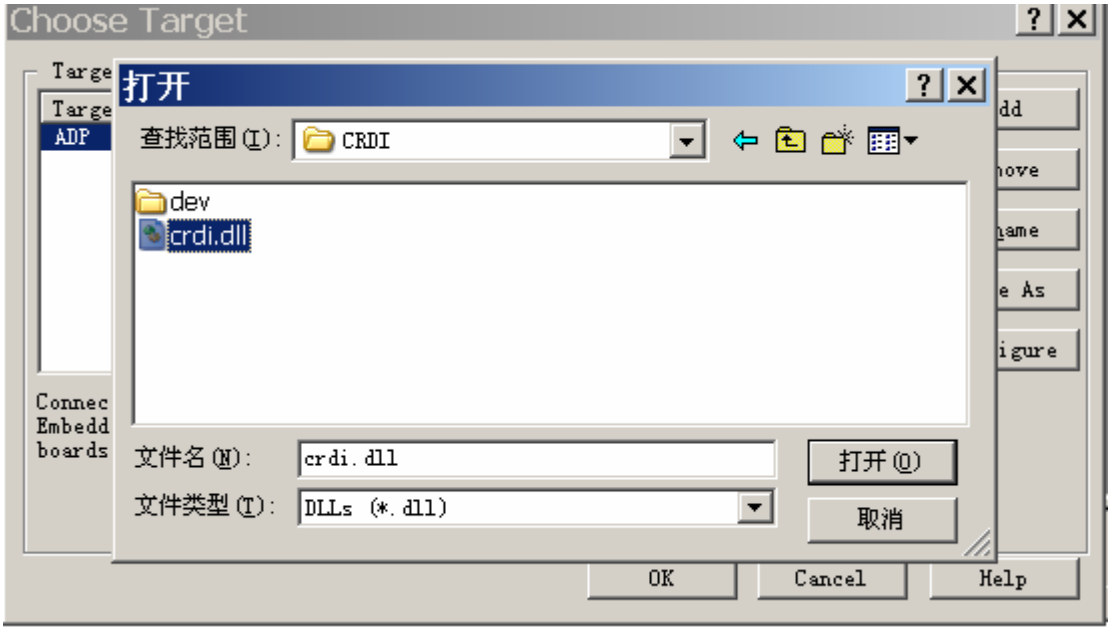


图 3-10

则 CRDI.dll 已经添加到 Target Environments 列表中,点击 Configure 进行配置:

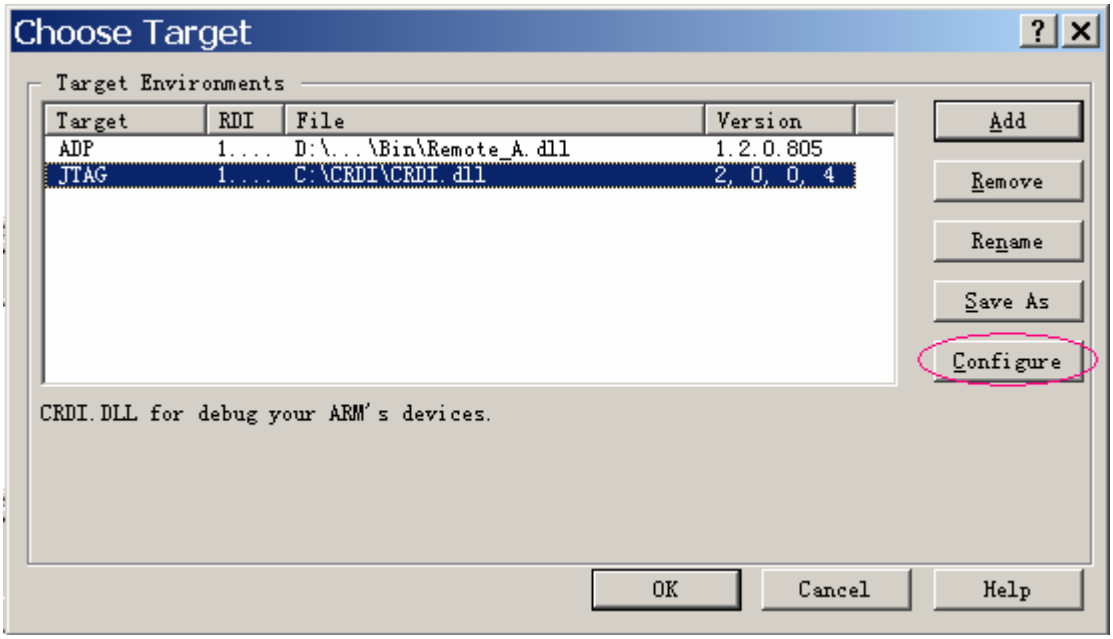


图 3-11

在如下窗口进行配置:

- RDI Setting

  nTRST enable

是否允许通过 nTRST 信号对 JTAG 进行复位，对调试无影响，可不选。

Turbo download enable

是否允许使用加速下载。当下载的程序较大时会明显减少下载时间。

Init chipset enable

是否在下载程序到目标板时对系统进行初始化。比如空板调试时可通过此项初始化 RAM 等.针对不同的目标板设定参见 Chipset Initialization 属性页的设定。

Endian

选择 Little endian

Catch Exceptions

是钟对 ARM9 内核进行中断捕获的,可忽略

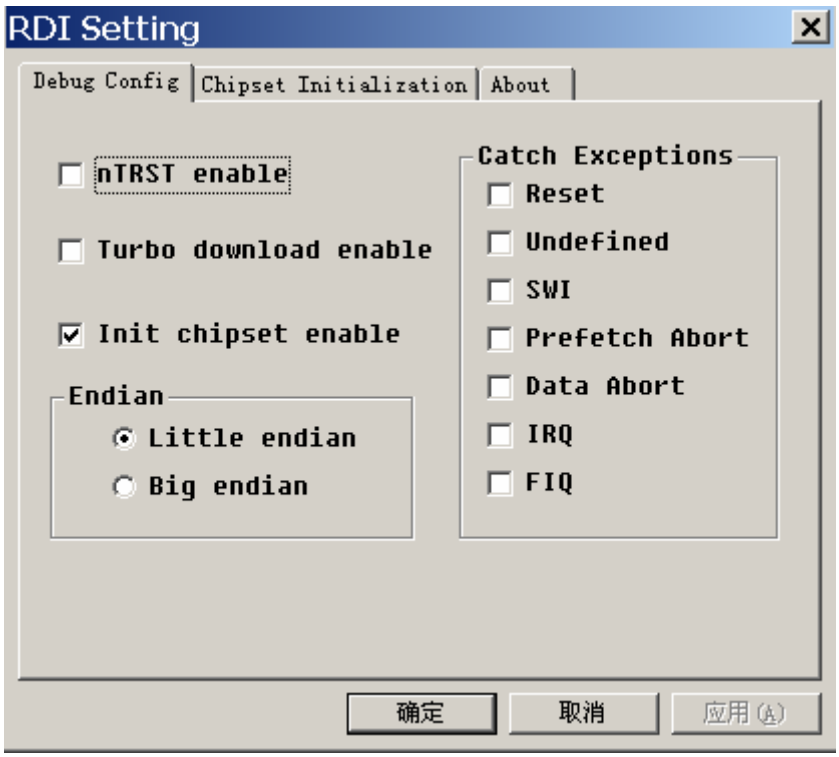


图 3-12

在 Chipset Initialization 页面选择 QT44B0X，如下图所示，点击确认关闭。

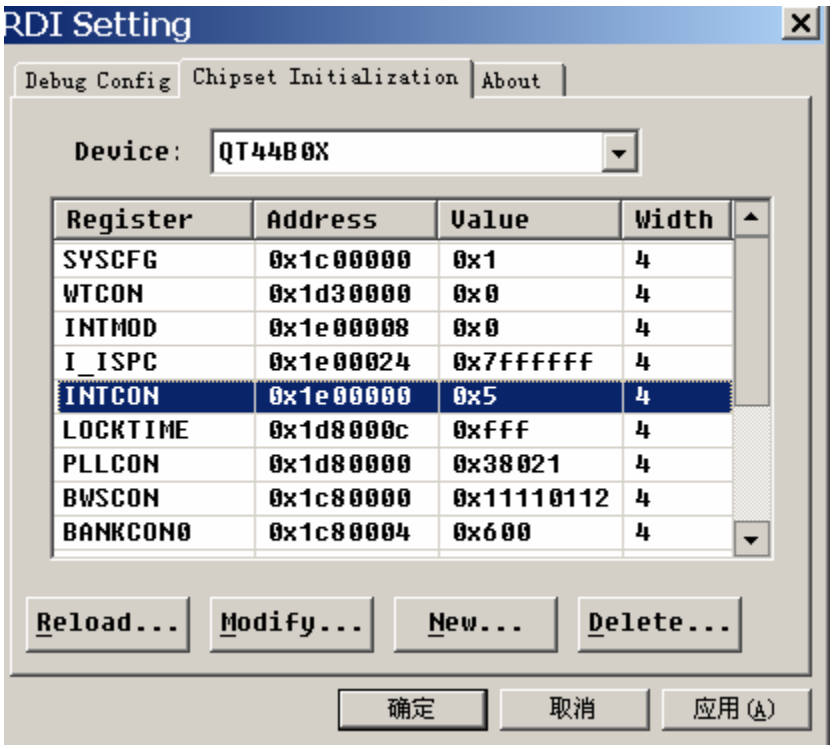


图 3-13

重新回到 Choose Target 对话框，点击 OK，然后关闭 AXD 即完成设置。

2）结合 ADS 进行仿真测试。

经过上文描述的步骤，我们已经得到一个可以仿真的环境，下面以 LED 跑马灯测试程序为例，详细介绍编译调试的过程。

打开 ADS，选择菜单 File -> New…创建新的工程文件

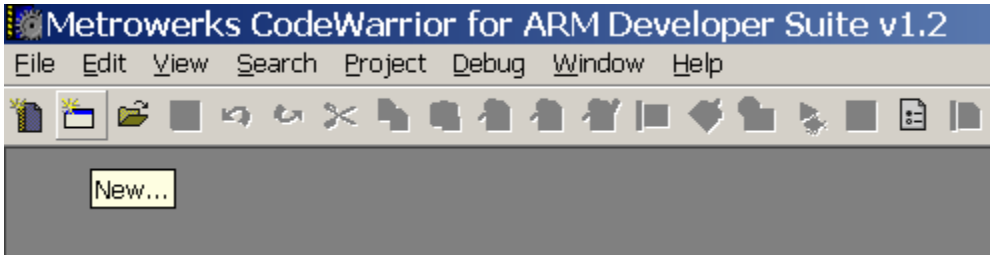


图 3-14

选择工程存放路径，输入工程文件名，如 LedTest

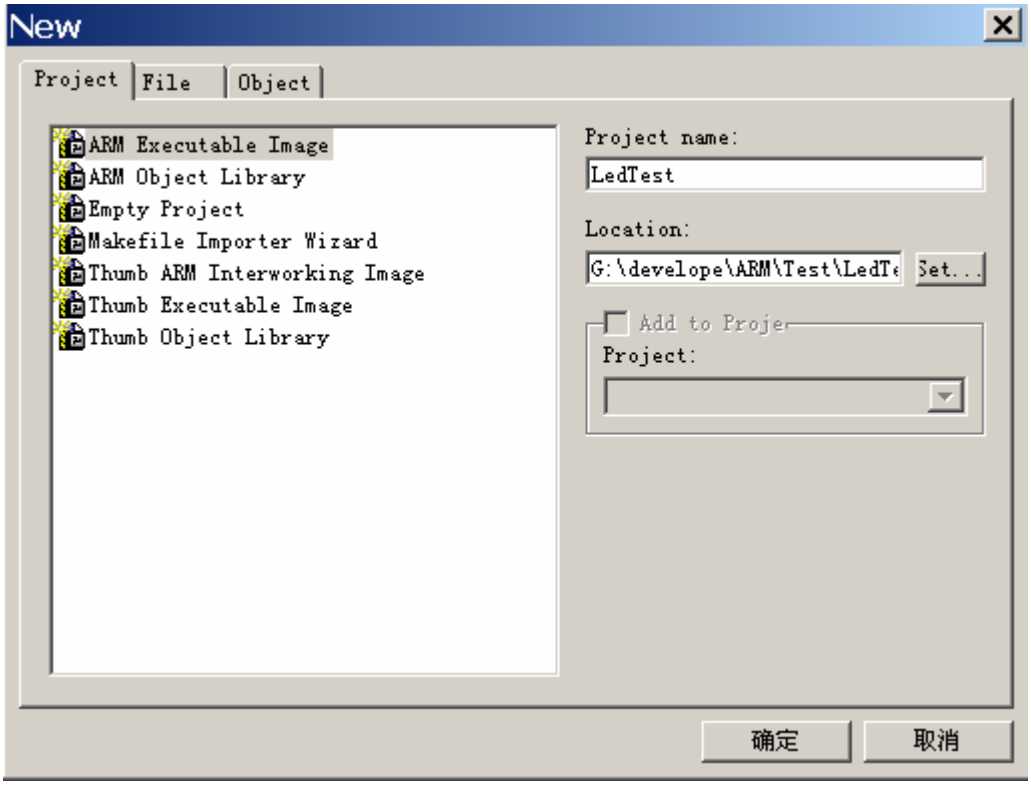


图 3-15

创建新的文件 LedTest.c， Iint.s，同时添加到新建的工程中

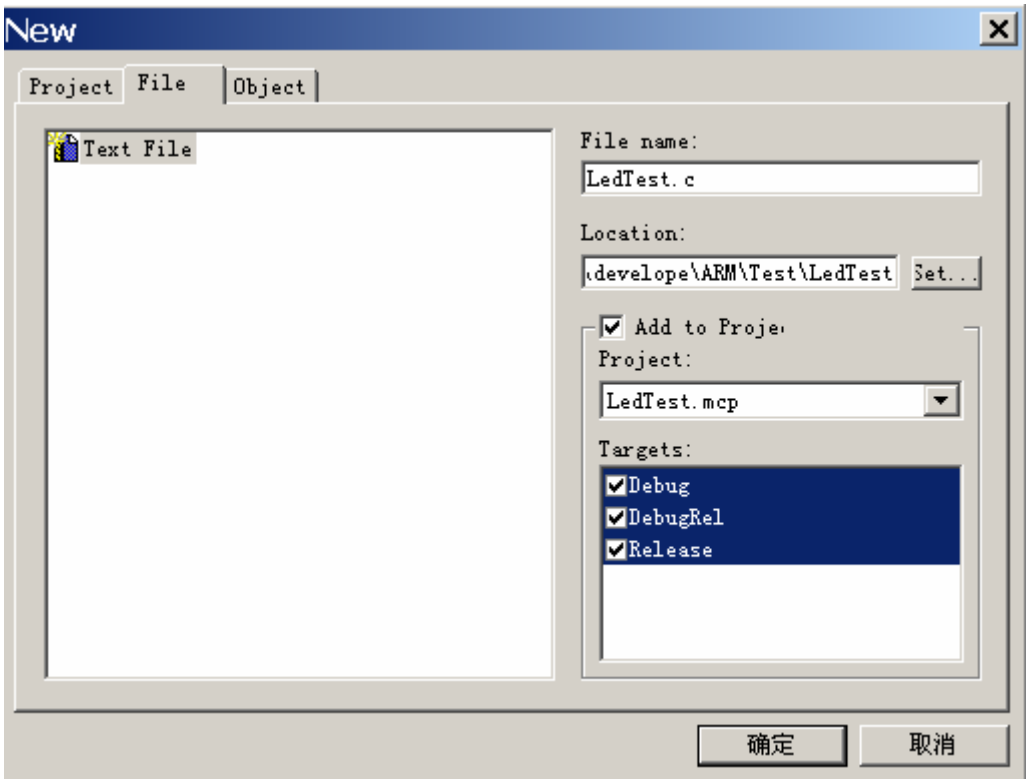


图 3-16

然后向文件输入相应的内容，内容略，请参考光盘 Source 文件夹中 LedTest 中的相关源代码。

文件输入完成后，需要对工程作一些设定以便在 QT44B0X 上运行。

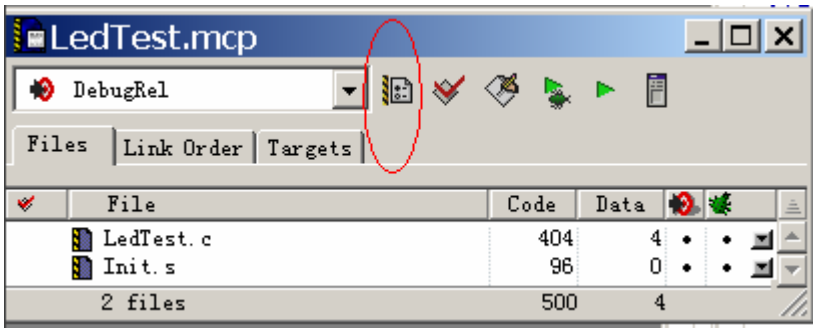


图 3-17

在 ARM Linker -> Output -> Simple image 中将 RO Base 指向 RAM 空间，如 0xC000000,以便编译后的代码可以写入到目标板中。

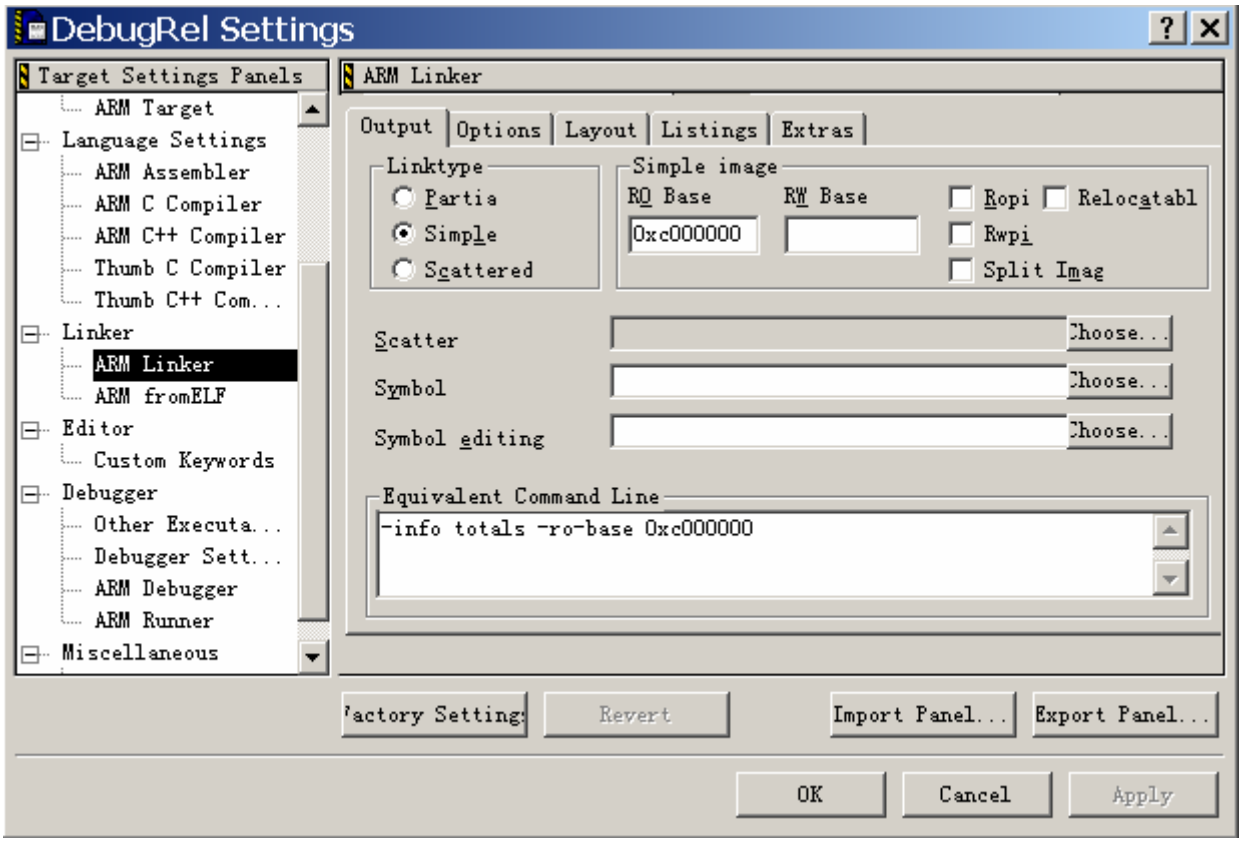


图 3-18

点击 Make 完成编译和连接，生成目标板可执行代码

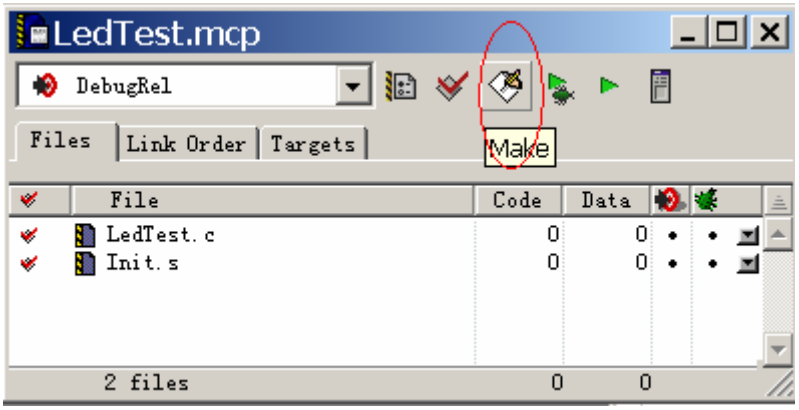


图 3-19

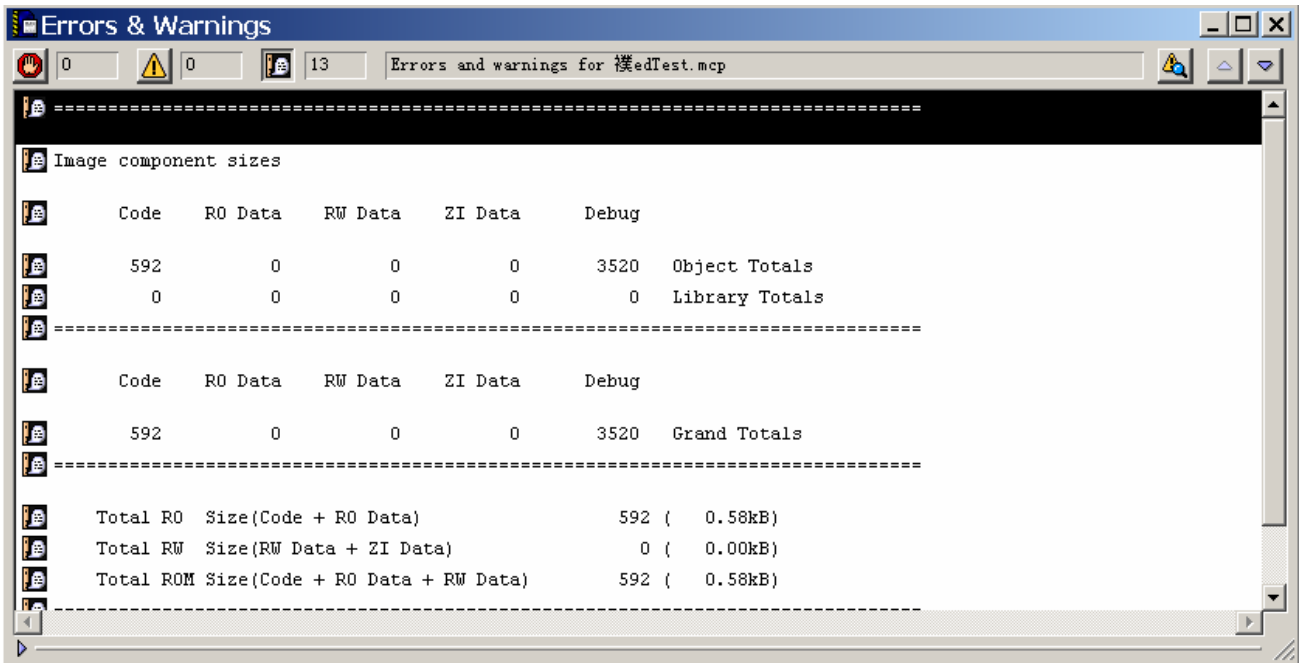


图 3-20

点击 Debug 开始调试

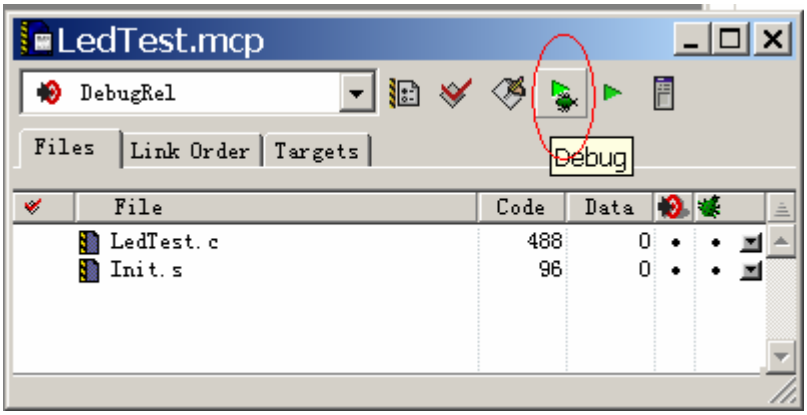


图 3-21

可以通过单步，步入，步出，运行到光标处或是使用断点来控制指令的运行，同时也可查看寄存器，存储器的数值。

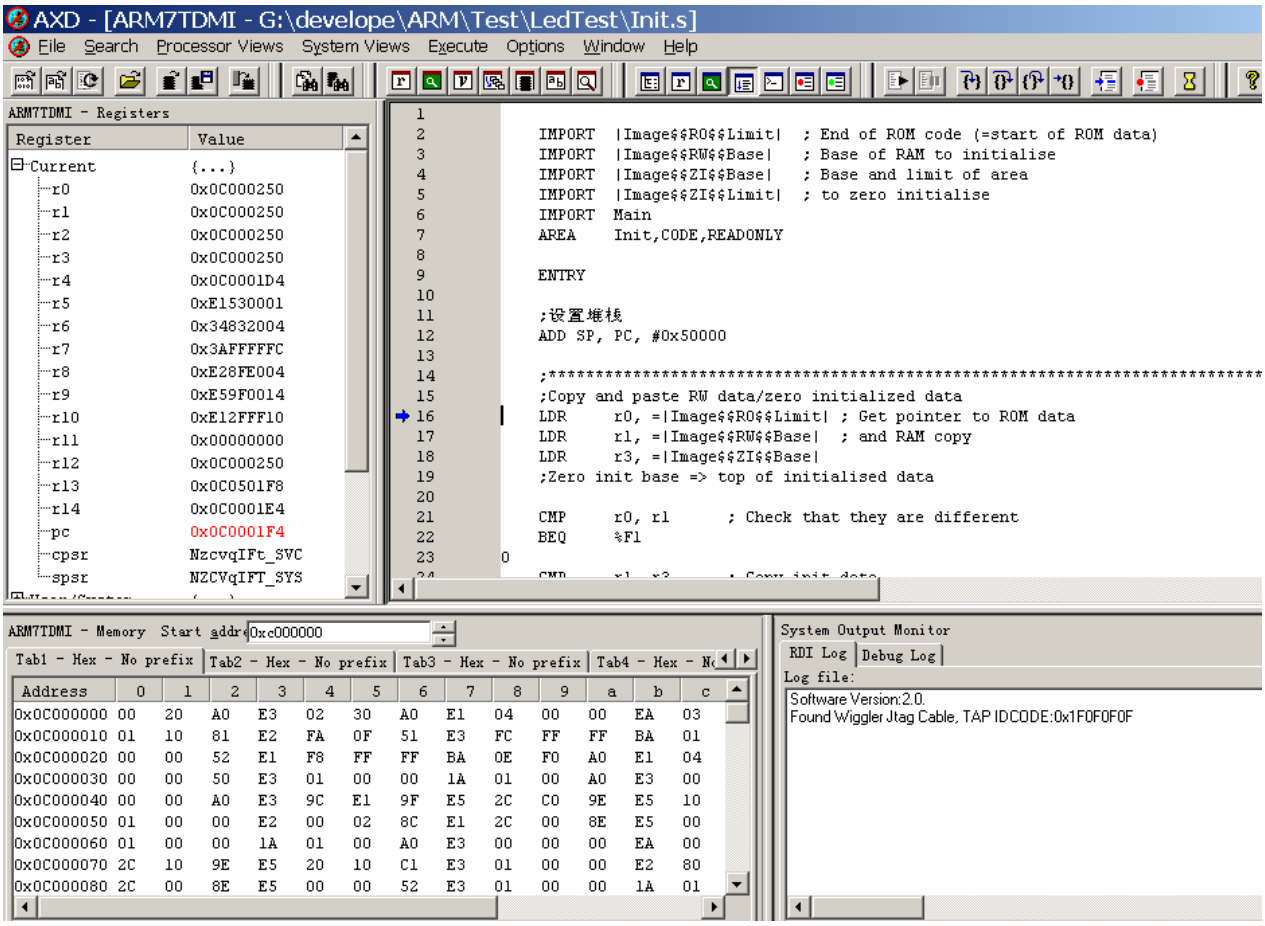


图 3-22

在右键菜单中选择 Interleave Disassembly 可在源码中显示汇编代码。

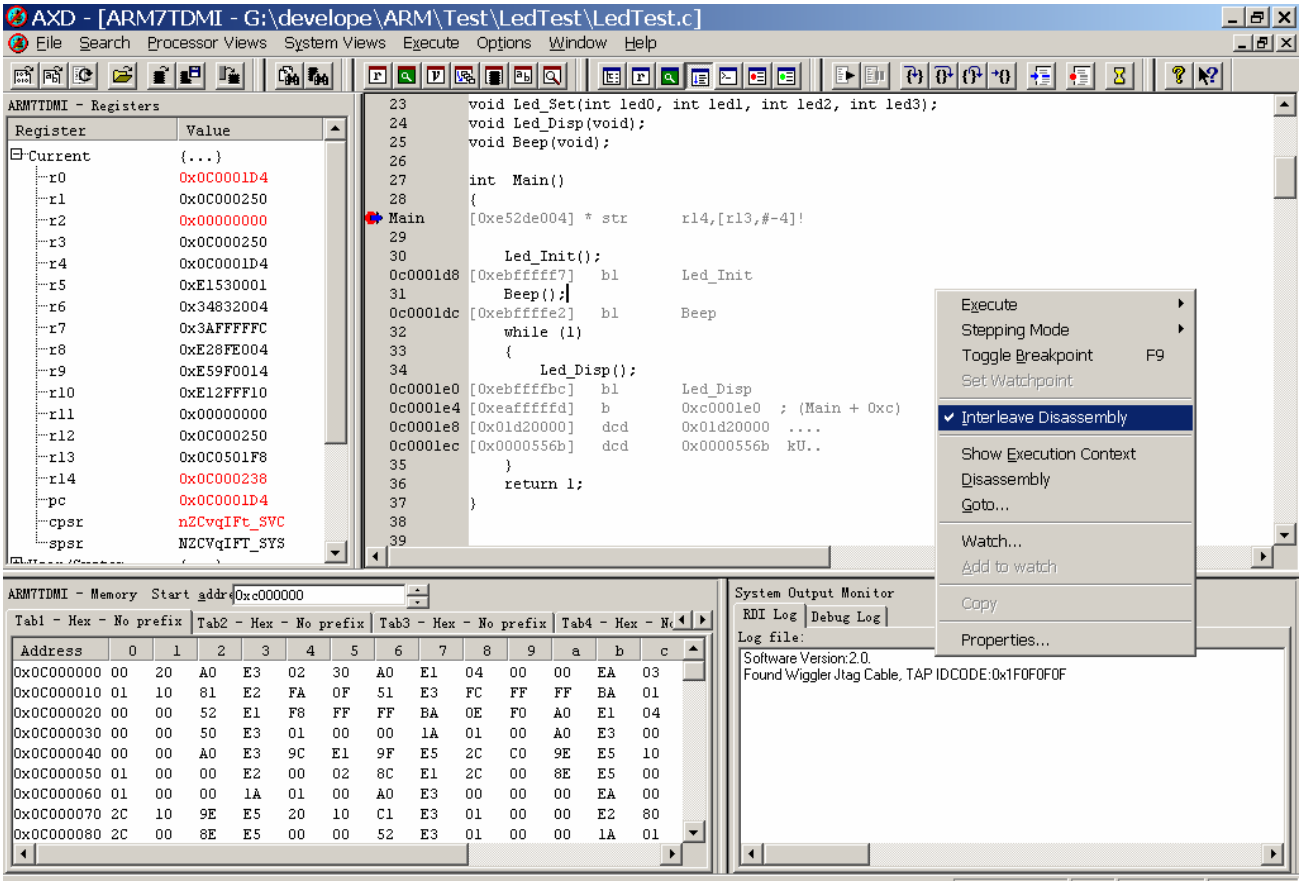


图 3-23

- 编译为 Thumb 代码进行调试

在 Settings 中 Target -> File Mappings，将扩展名.c 的编译器修改为 Thumb C Compiler

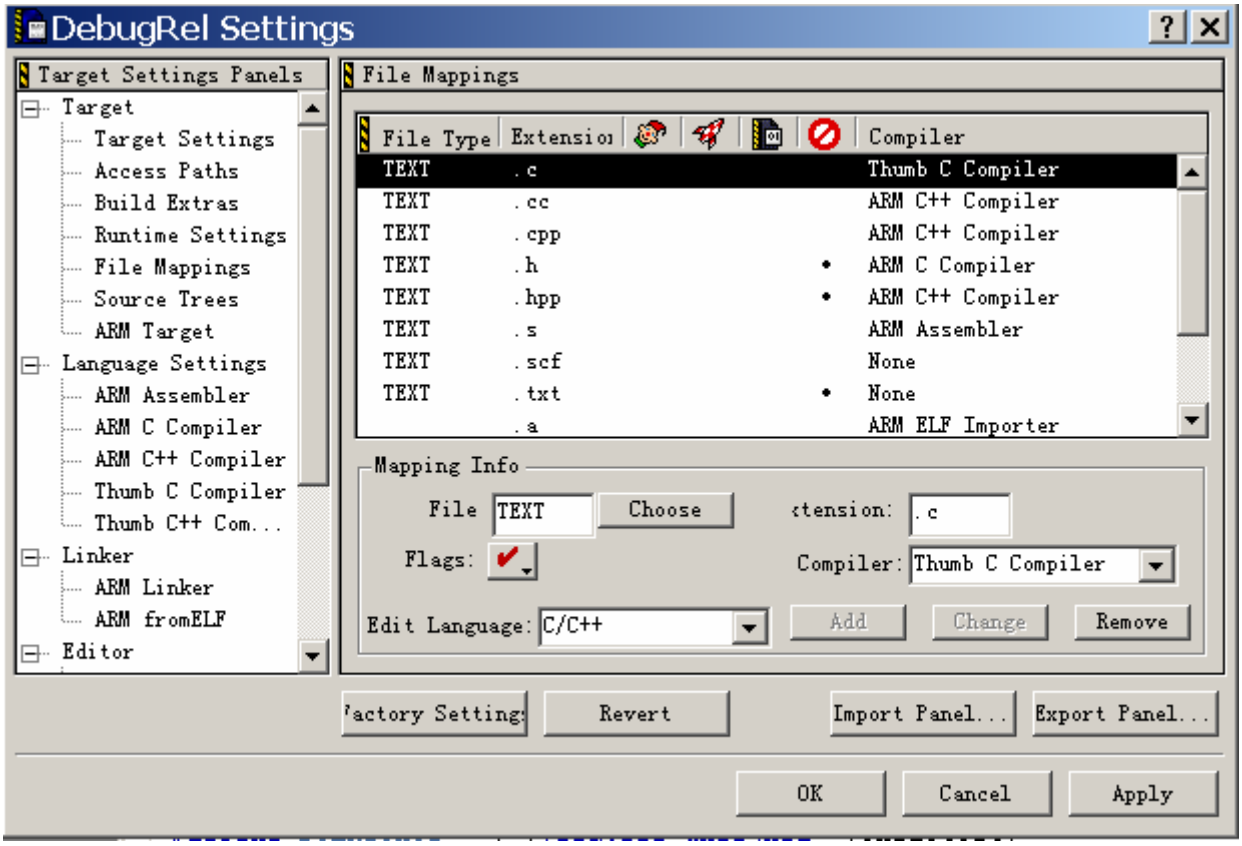


图 3-24

重新进行 Make，C 代码被重新生成 Thumb 代码。调试方法同上，只是生成的代码长度会短。

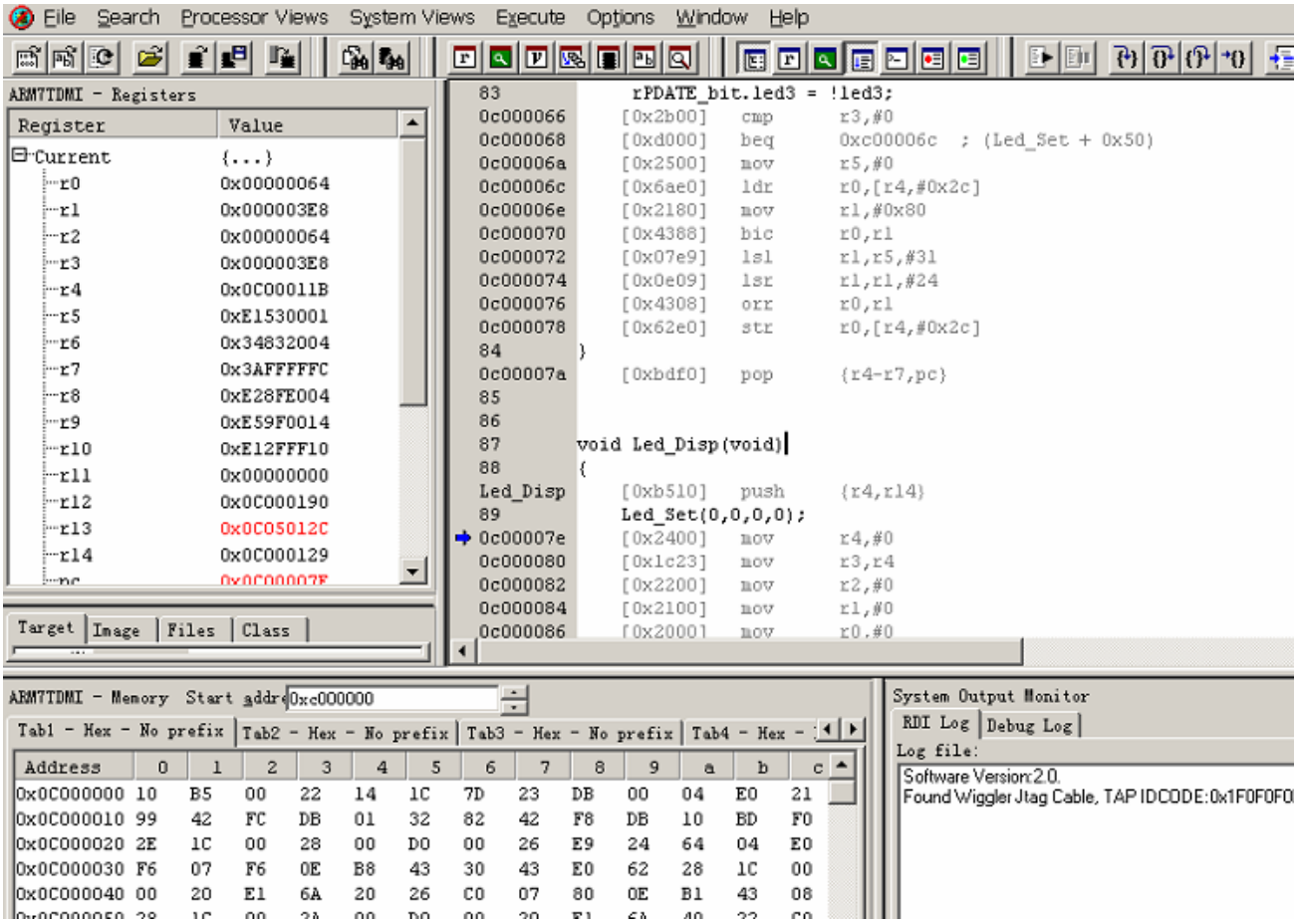


图 3-25

3、使用 Smart-ICE 和 ADS1.20 环境下进行仿真调试

1) Smart-ICE 仿真器的安装

- 安装 Multi-ICE 的驱动程序

  驱动程序位于配套光盘中 Tools\Smart-ICE 文件夹下

- 把 MULTI-ICE 和开发板连接，接通电源

- 启动 Multi-ICE server

  启动界面如下图所示

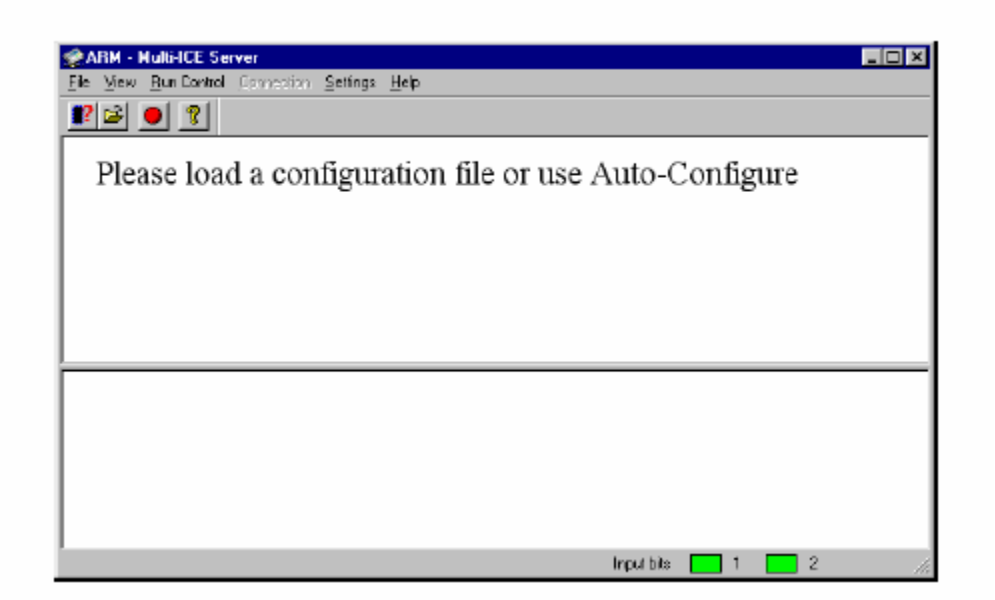


图 3-26

然后进行 configure，使用自动配置。

File -> Auto-configure，或者点击图标

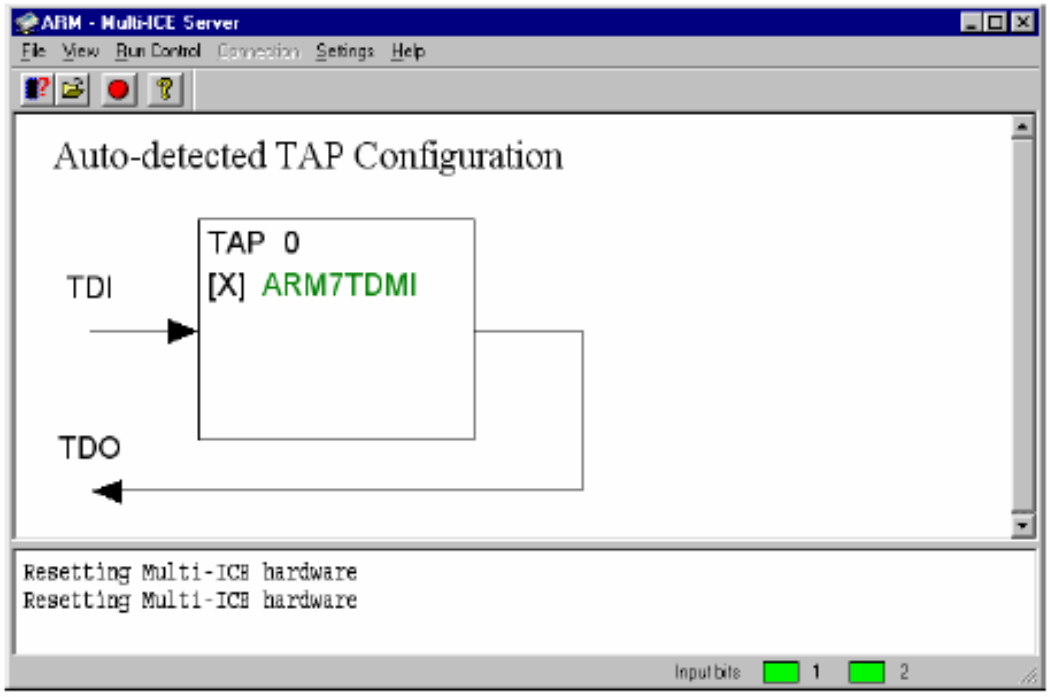


图 3-27

出现如上图所示的提示信息后，ICE 仿真器与开发板连接成功。

2）用 Smart-ICE 仿真器在 ADS 1.20 中进行调试

启动 ADS1.2，打开工程文件 LedTest.mcp

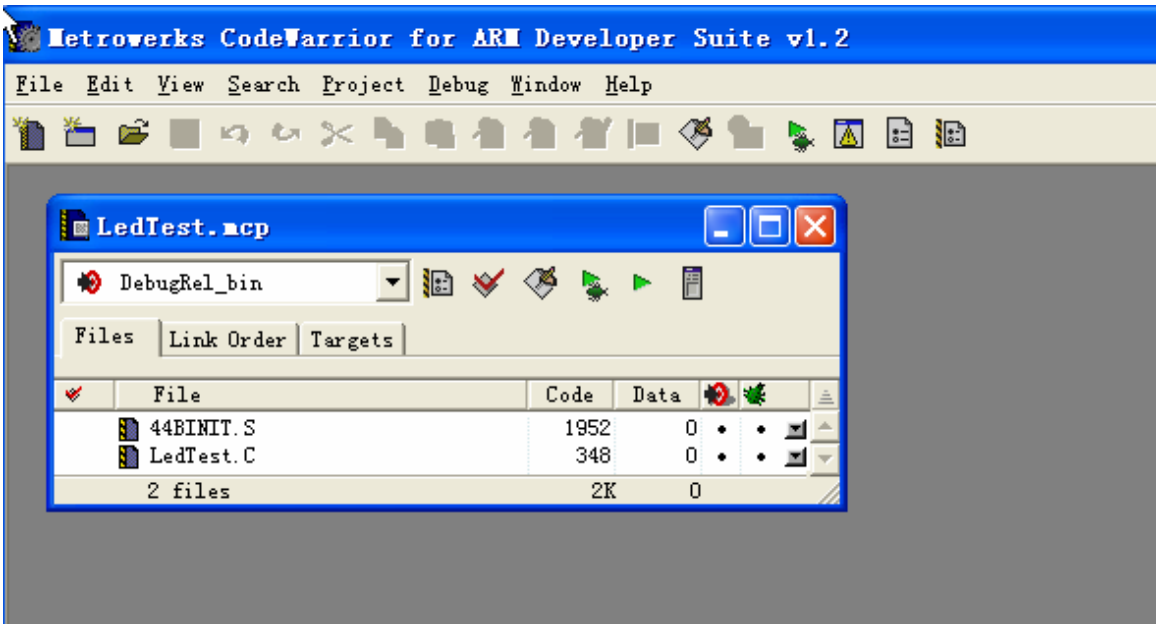


图 3-28

点击图标  进行在线调试。AXD 窗口将打开。第一次使用 AXD 要进行配置，配置方法在菜单 Options->configure target

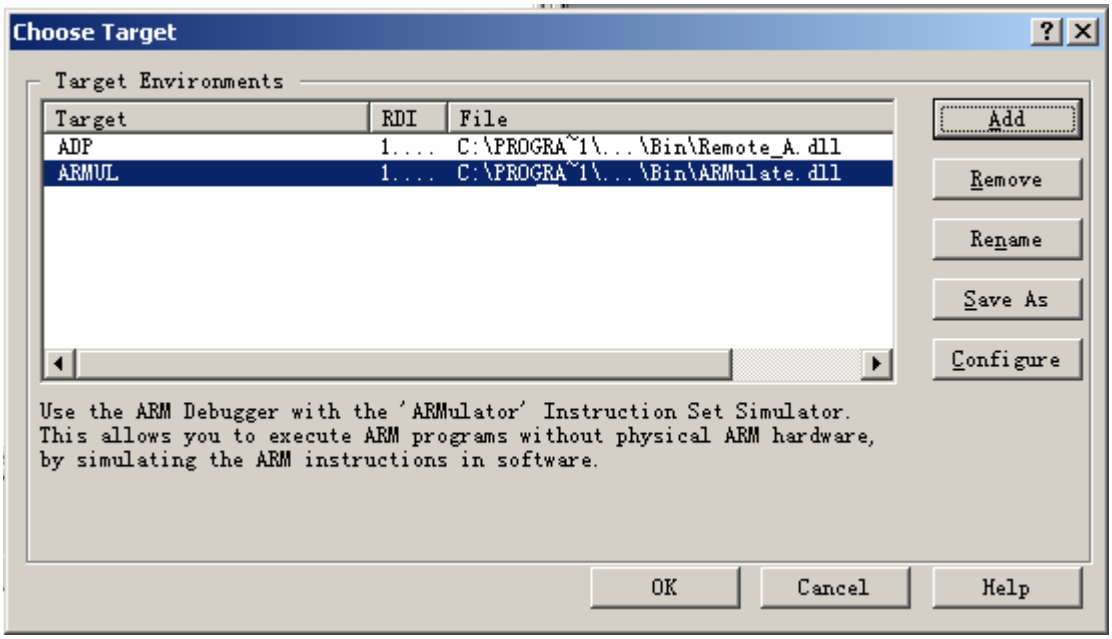


图 3-29

点击 Add 按钮，弹出浏览窗口，选择安装 Multi-ICE SERVER 的目录，选择

Multi-ICE.dll，如下图所示：

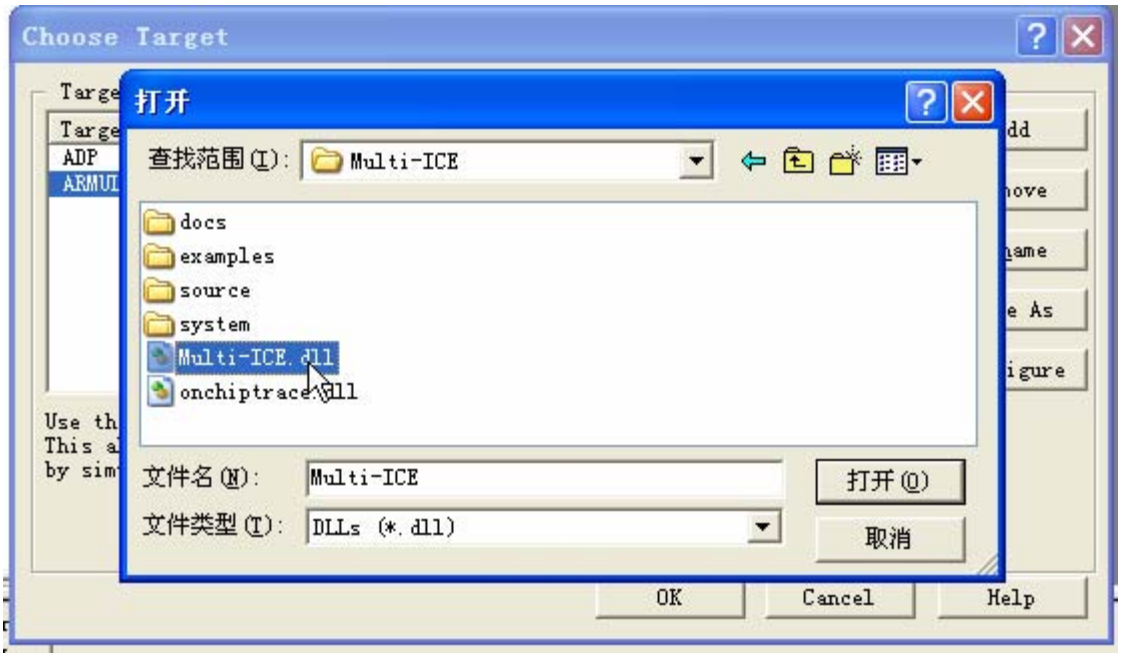


图 3-30

点击“打开”，“OK”按钮会会看到提示：

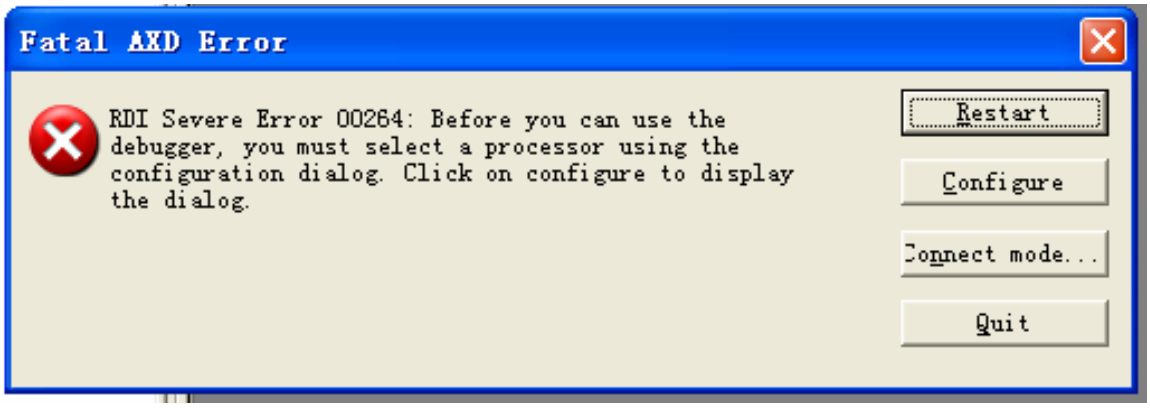


图 3-31

点击 Configure 按钮，出现下面界面：

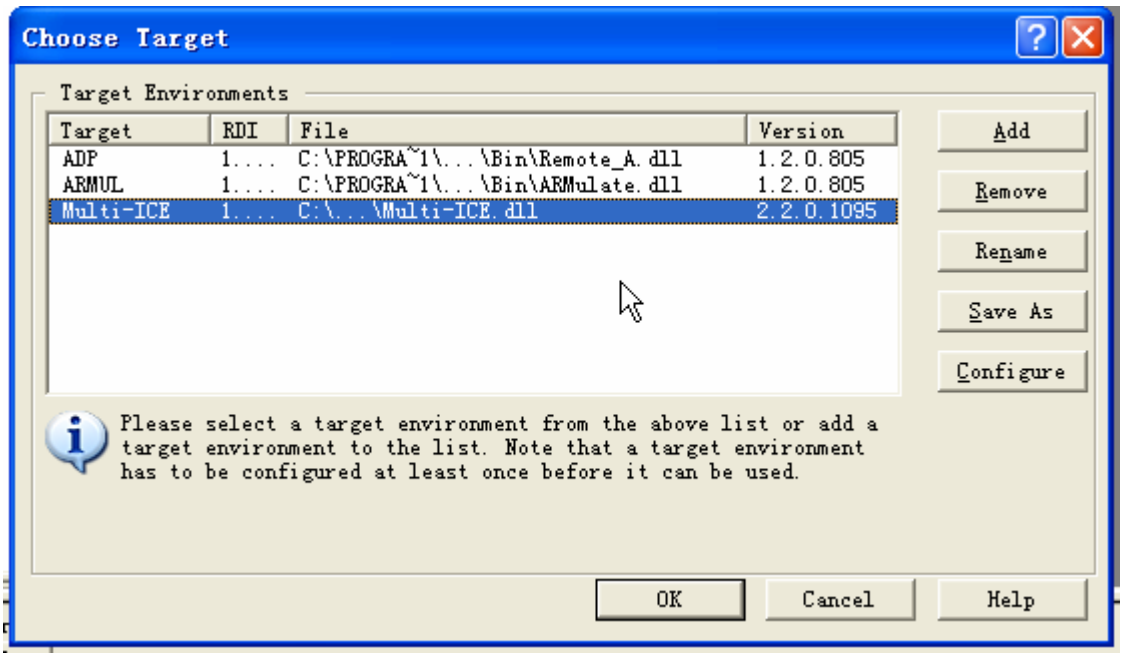


图 3-32

再点击 Configure 按钮，

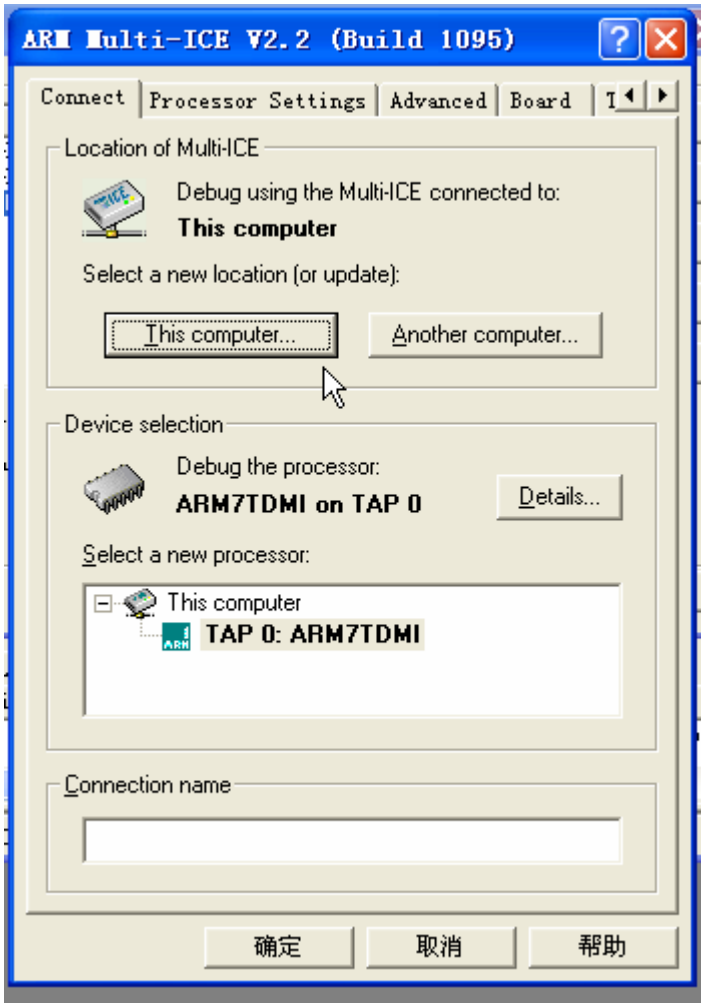
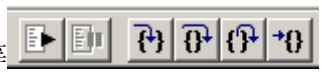


图 3-33

点击 TAP 0:ARM7TDMI 然后确定。

然后退出 AXD，在 ADS 下再次点击 DEBUG 按钮，这次进入 AXD 就不会出现错误。

启动 AXD 后，需要对存储器空间进行初始化，初始化命令的文件在配套光盘 Tools\Smart-ICE\44b0x.ini，为了方便使用，建议将该文件复制到 C 盘根目录下。在 ADS 环境下按 ALT + L 打开命令窗口，输入指令 obey c:\44b0x.ini 进行存储器的初始化。

然后从新 LOAD 文件，点击图标  ，或者菜单 File—>Reload current Image 这样就可以开始调试了，单步，全速，暂停等  。

在光盘中，我们提供了 ICE 仿真器使用方法的动画演示。

4、输出在 ROM 中运行的文件

程序调试完通常要写入 ROM 中然后上电运行。需注意如下事项：

1）需要加入初始化代码。在上面调试器调试时，一部分初始化的操作已经由调试器完成。系统上电时，如 RAM 未被初始化不能直接的使用，需要在代码中添加相应代码，然后初始化堆栈、外设，代码和数据拷贝，然后跳转到 C 代码入口开始运行。

2）重新定义代码的映像地址。在调试中，通常将 RO 和 RW 均指向 RAM 地址，此时要分别指向 ROM 和 RAM 地址。重新设定下图中 RO Base，RW Base 的地址。

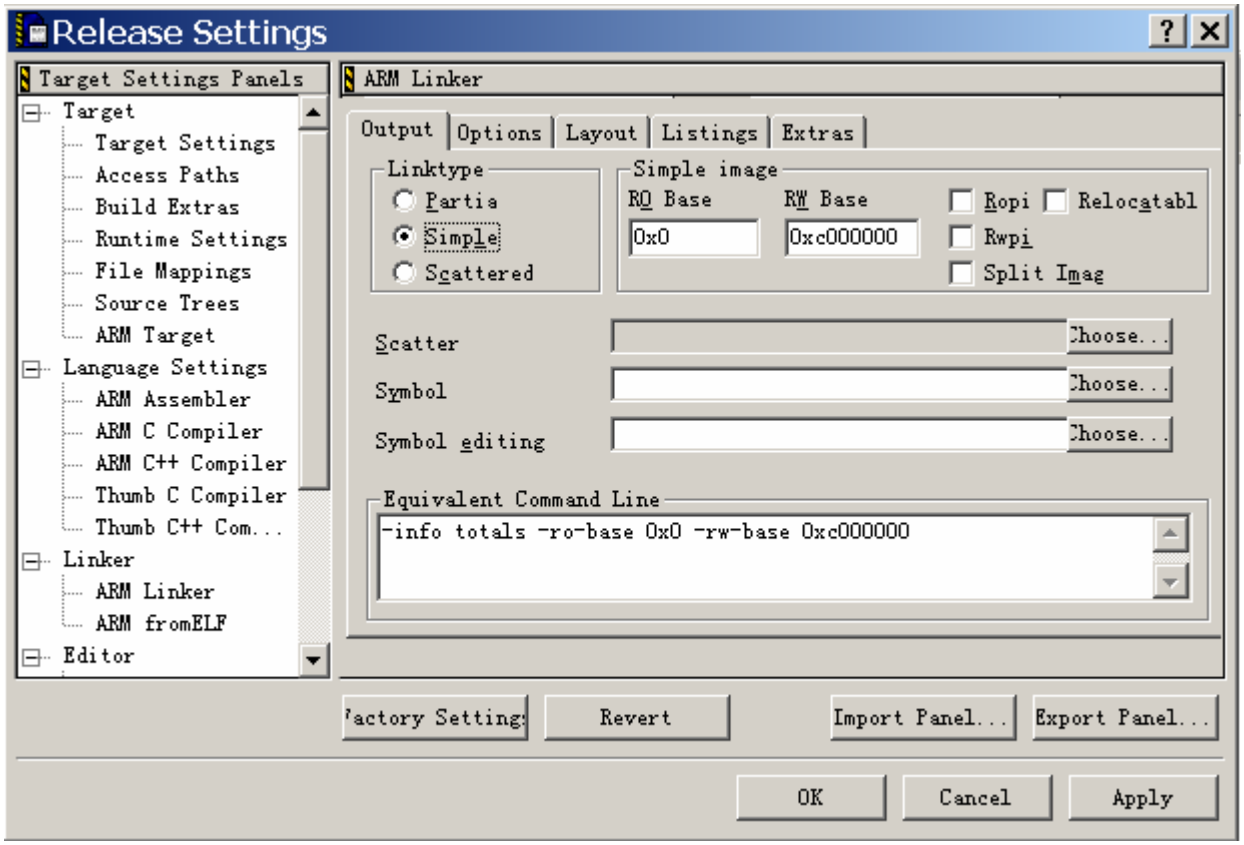


图　　3-34

上面的调试在调试过程中，调试器会自动调整 PC 值指向 ENTRY 标识的入口处。而系统上电从地址 0 开始执行，所以须要将 Init 段置于映像文件的最前面，参考如下设置，使 0 地址指向 ENTRY 标识的入口处

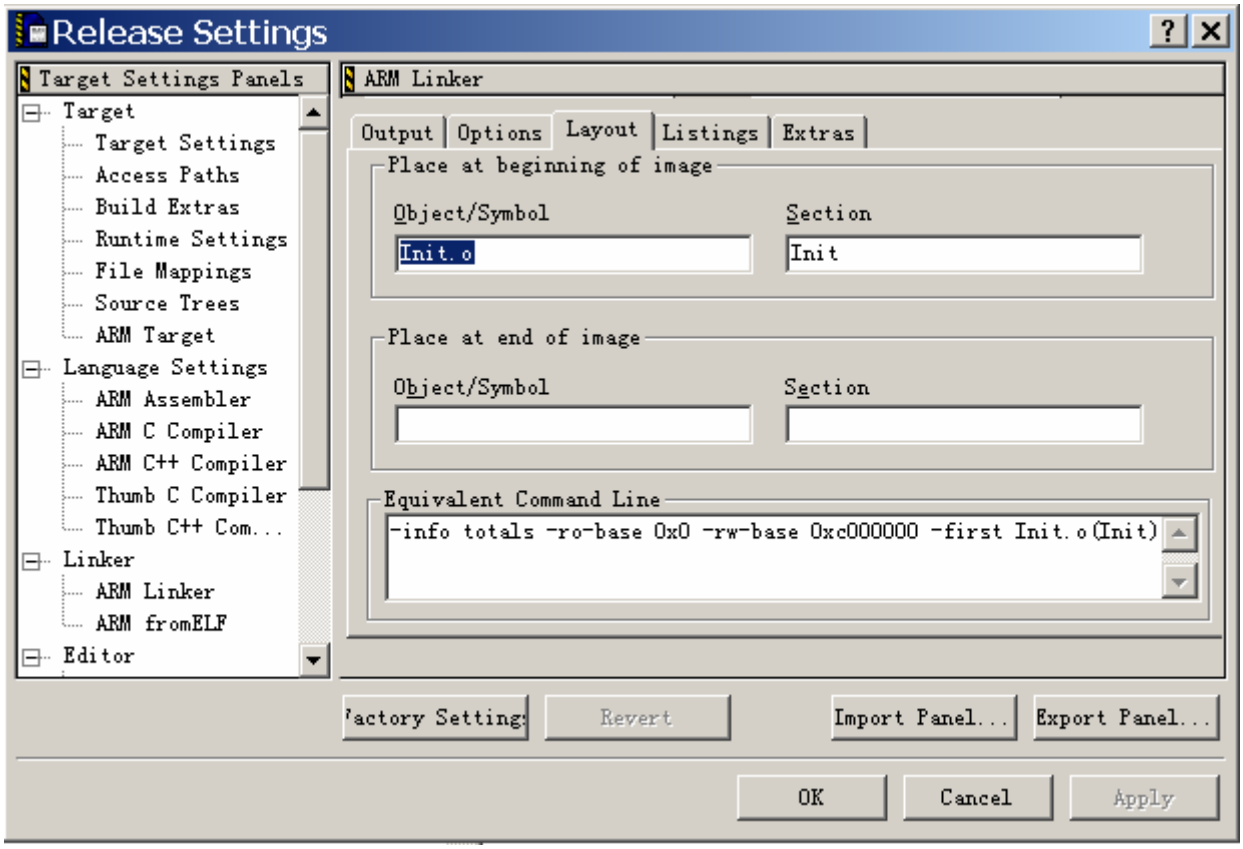


图 3-35

选择输出可执行代码的存放格式（如下图所示），再次编译，就会生成文件，烧写方法请参考下一章《如何快速烧写 Flash》

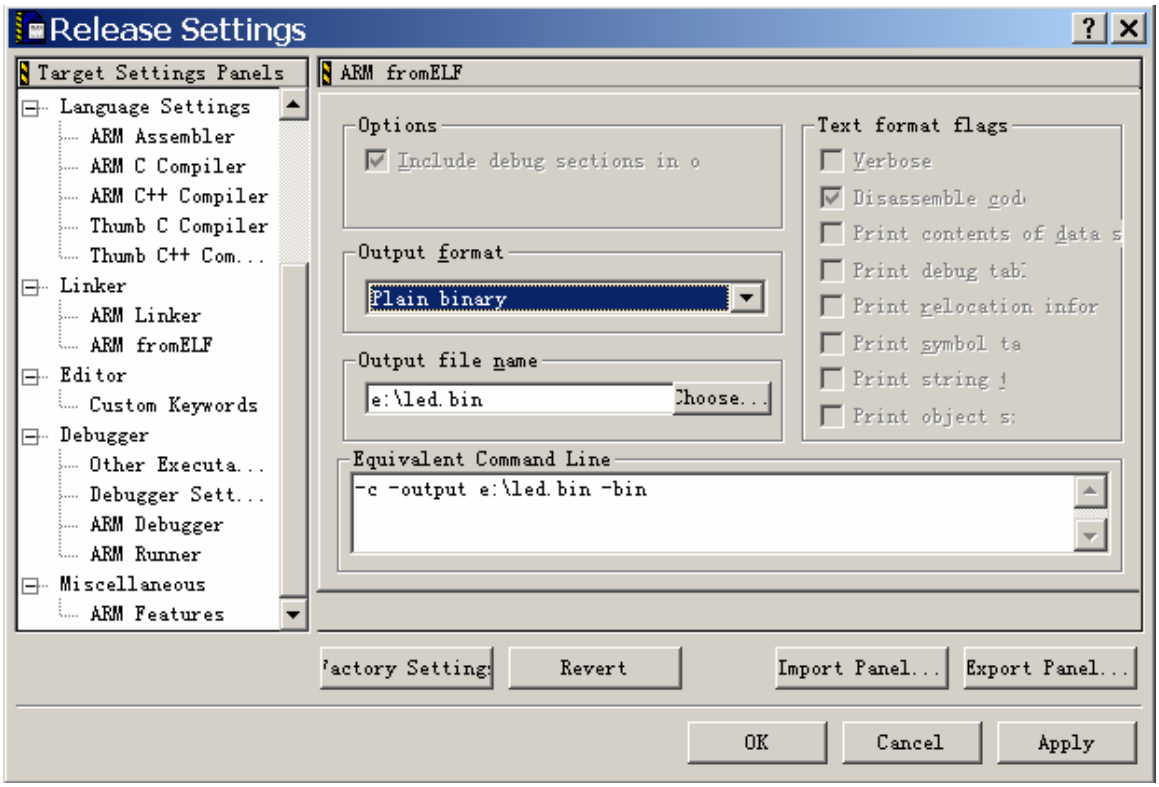


图 3-36

5、其它调试方法

除了上述两种调试方法外，还有许多别的调试方法，用户可以根据个人喜好，选择最适合的调试方式。

在 uClinux 的环境下，我们可以通过 arm-gdb 来进行调试，调试方法请参看本说明书 uClinux 部分。

第四章　如何快速烧写 Flash

1、计算机的设置

请在 PC 机的 BIOS 里面将并口设置成 EPP 方式（如果是 IBM 的笔记本，那么请参考 IBM 网站找到你本本的型号进行相应的设置）；

2、ELF 文件的准备

flashpgm不能烧写*.bin 格式的文件,但可以烧写*.S19、*.elf或者是*.hex 格式的文件，一般都使用*.elf 格式的文件进行烧写。

以 ADS1.2 为例，在其项目编译界面下按 ALT＋F7 进入下面的设置界面，按照图 4-1 中画红色圆圈的 Target Settings 目录下选择 ARM fromELF，以生成目标代码：

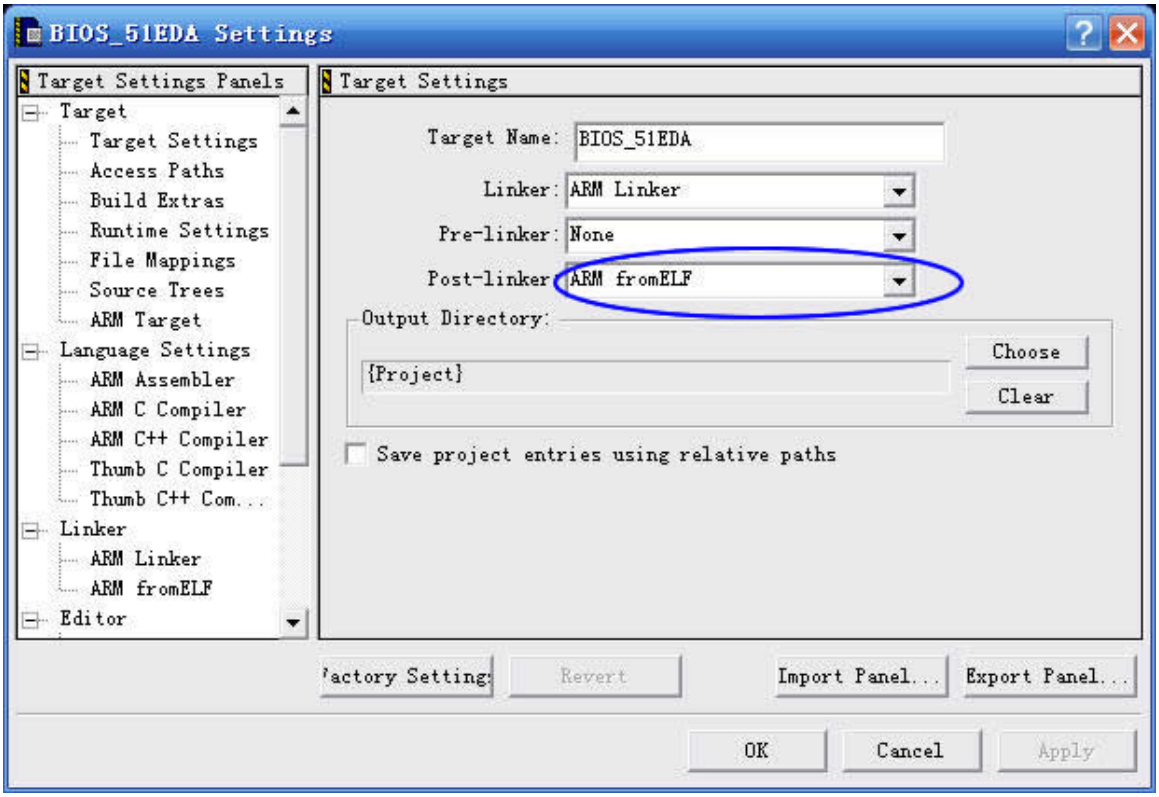


图 4-1

然后在左边栏中 Linker---ARM formELF 菜单下的 output format 选择 ELF,输出任意文件名与 ELF 后缀。

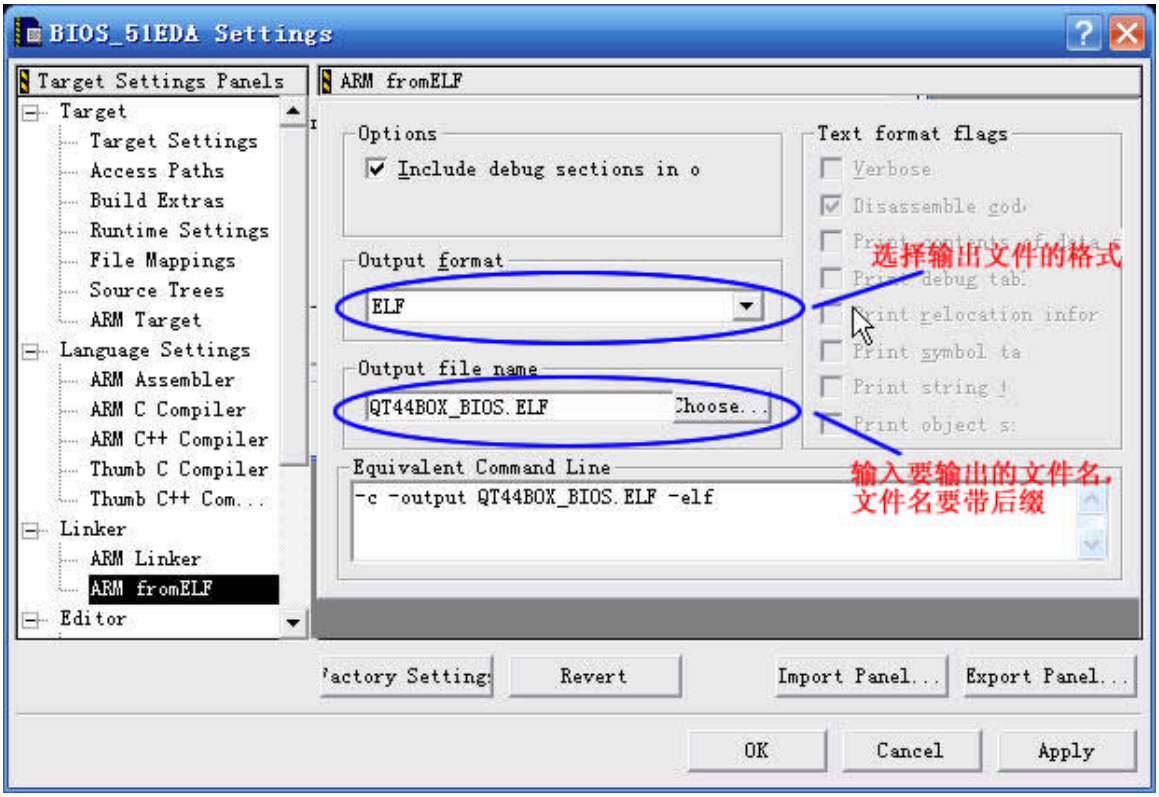


图 4-2

然后编译就会生成你所设置文件名的 ELF 格式文件了。

3、Flashpgm 烧写前的准备

联接主机、开发板和烧写电缆，JTAG 电缆推荐接到子板的 Wriggle 口。

4、FlashPgm 的设置

调入 QT44BOX.ocd 文件，该文件存放在配套光盘 Tools\OCD 文件夹中，见图 4-4

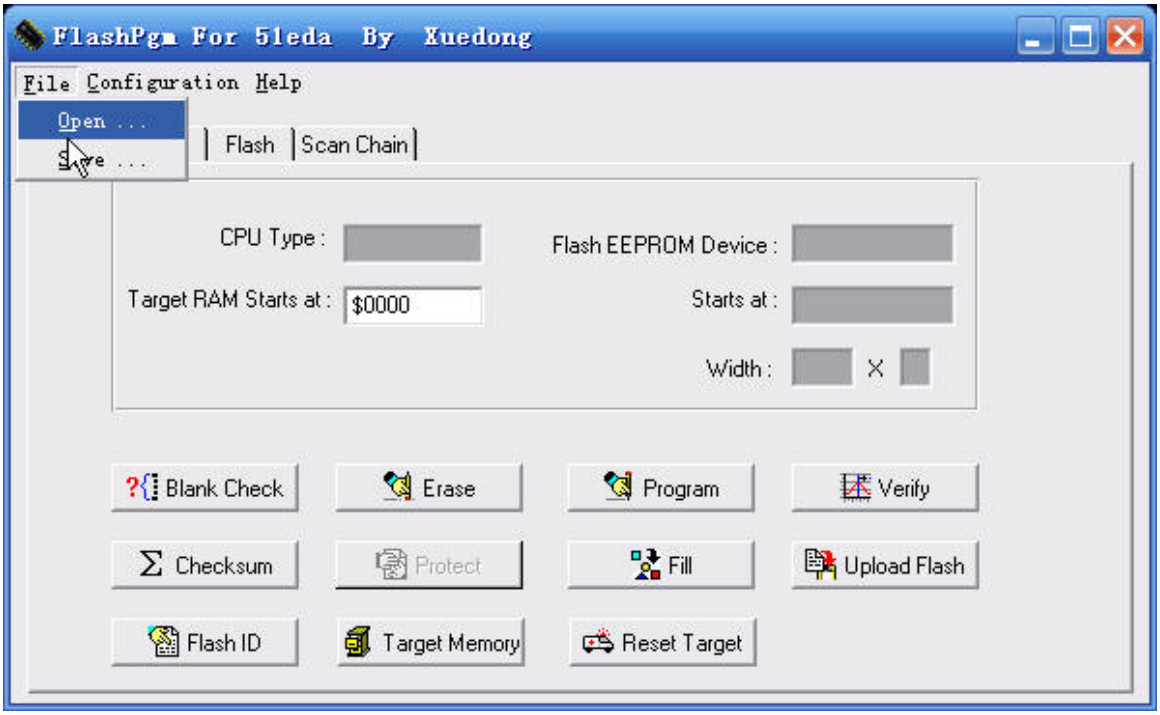


图 4-3

选中我们提供的 OCD 文件，见图 4-5

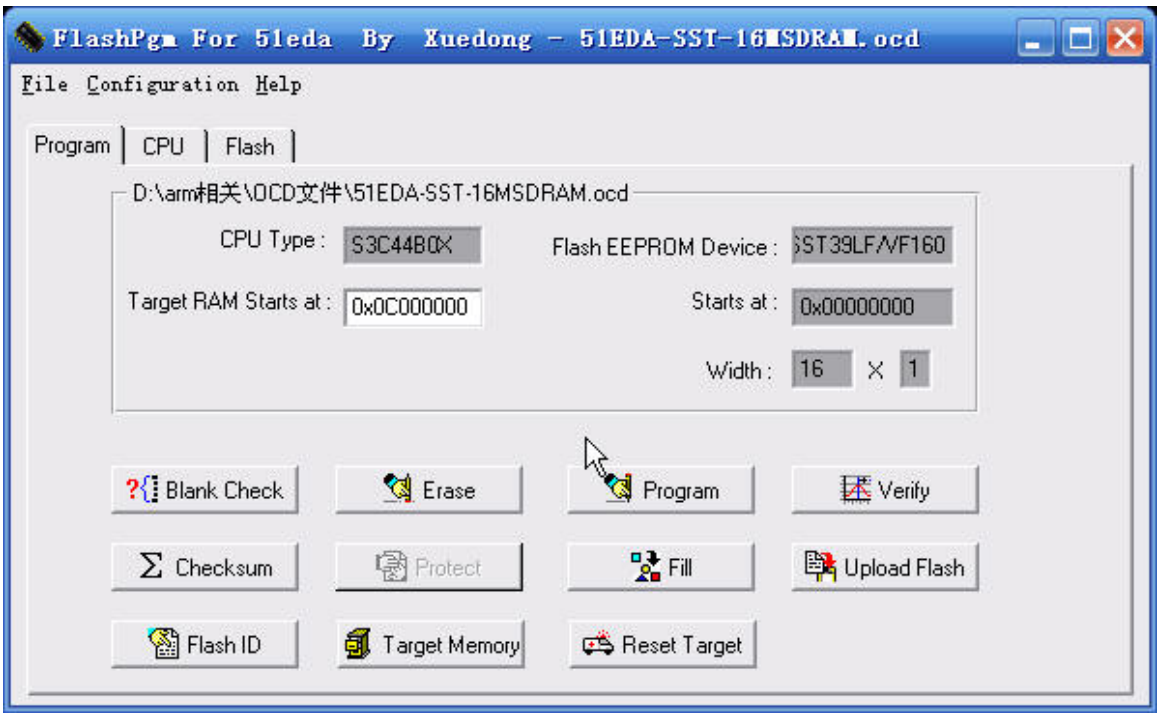


图 4-4

设置通信端口（见图 4-6）

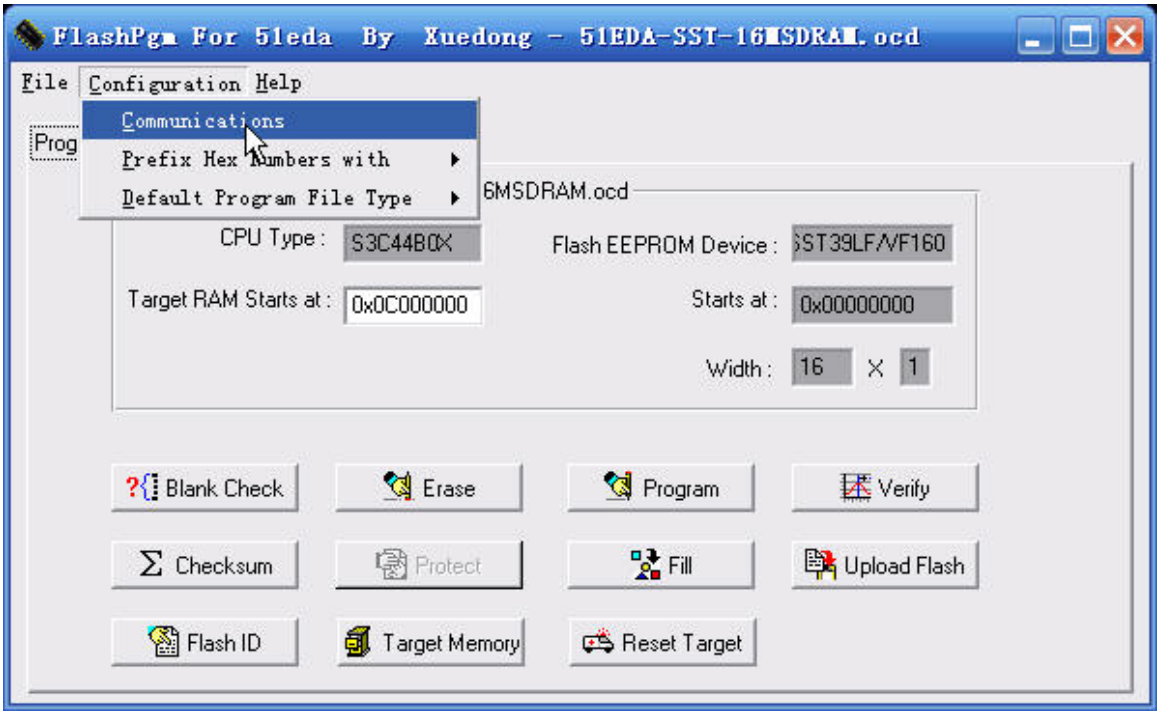


图 4-5

选择 Wiggler Parallel（见图 4-7）

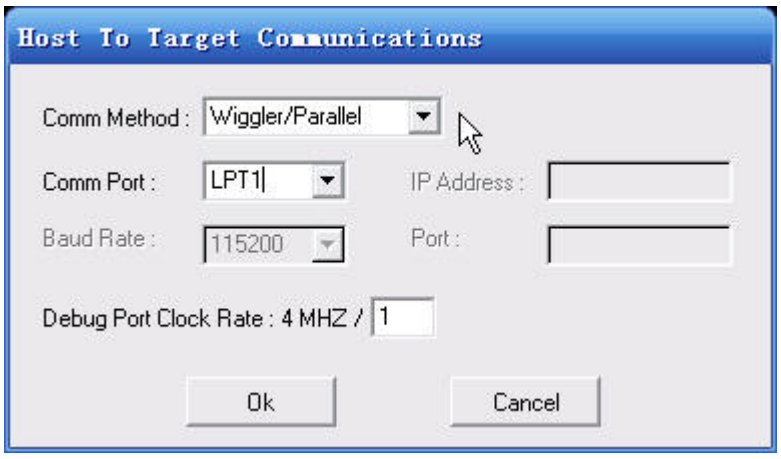


图 4-6

选择默认的烧写文件类型（见图 4-8）

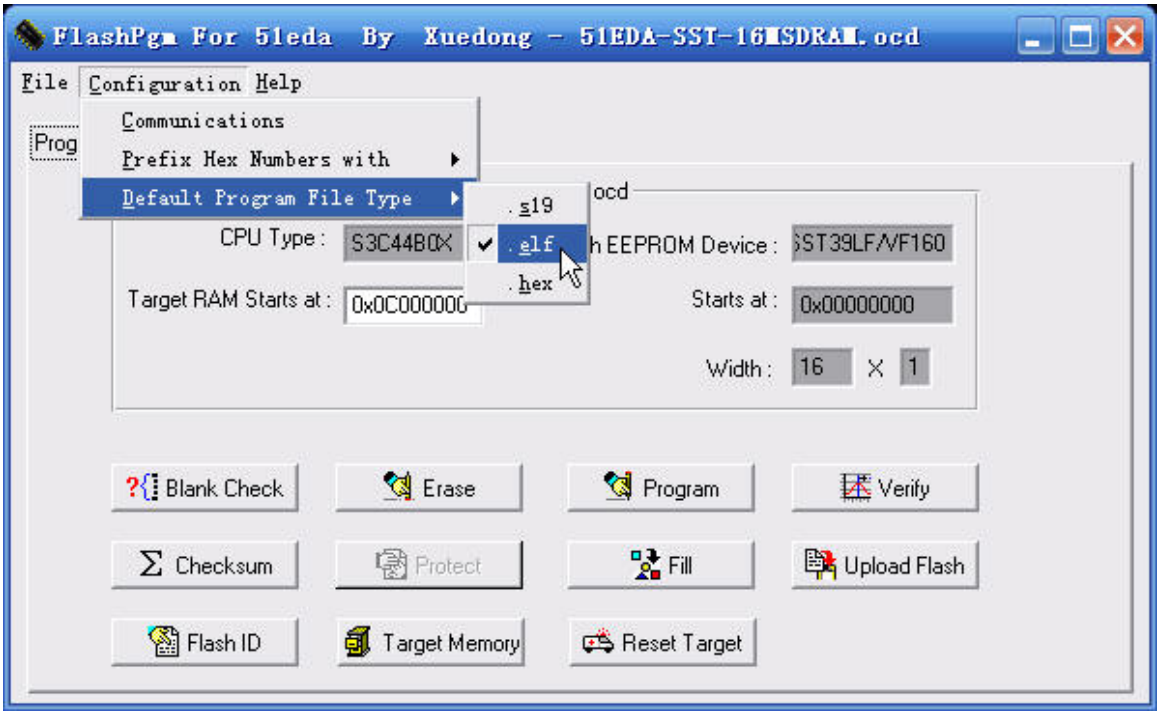


图 4-7

点击编程按钮（见图 4-9）

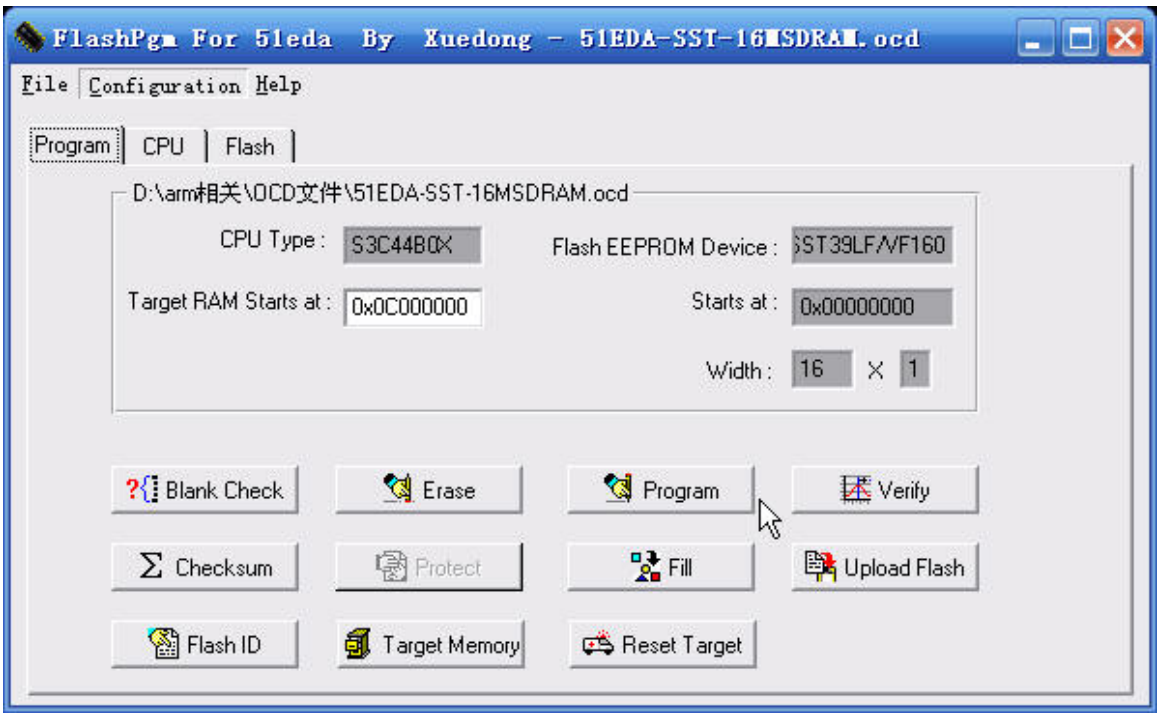


图 4-8

如果目标板连接不正常（见图 4-10），请检查目标板电源是否打开，20 芯排线

是否连接正常。

图 4-9

如果连接正常（见图 4-11）

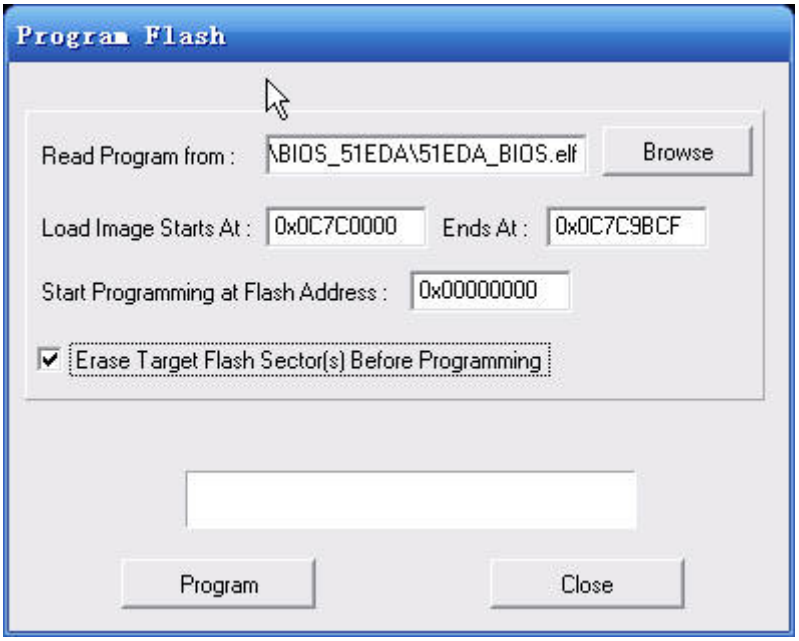


图 4-10

点击 Browse 按钮，找到要烧写的文件的位置，另外，我们也要选中 Erase Target Flash Sector(s) Before Program

如果在上面的画面中，你发现按键 Program 始终为灰色，无论如何也不行，或者在烧写过程中出现烧写失败（见图 4-12）请点击 Close 按键关闭上面的画面，然后按一下板子上面的复位按键，再点击 Flashpgm 界面中的 Program 按键，重新进入上面的画面，就不会出现上面的错误情况，这样就可以编程了

注：Program 始终为灰色是 Flashgpm 的一个 bug；烧写过程中出现烧写失败是

因为 Flashgpm 不支持 SST39VF1601 的芯片，直接用 SST39VF160 代替。

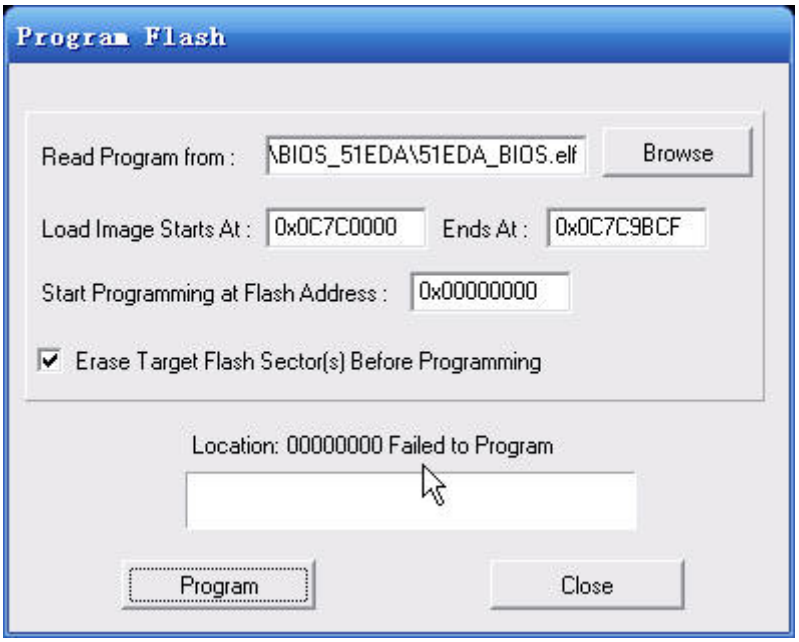


图 4-11

烧写成功以后会出现”Program/Verify Complete”的成功提示。（见图 4-13）

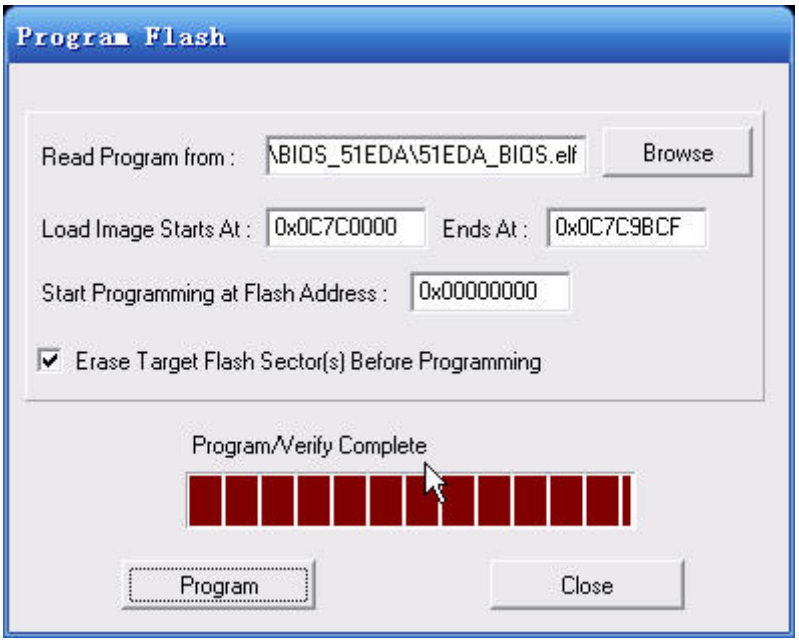


图 4-12

烧写完成。

第五章　QT44B0X-I_BIOS 的烧写与使用

1、BIOS 的编译与烧写

1)　通常我们不用重新编译 QT44B0X－I_BIOS, 如果要编译，那么请使用配套光盘中 Source\ 51EDA_QTBIOSV1.0 文件夹下的源代码，这是一个 ADS 1.20 的工程，请先参照本说明前面的章节安装 ADS 1.20。

2)　设置 ADS 1.20 的编译环境。

如图 5-1 所示，点击红框处的按钮将弹出工程编译设置界面图 5-2，在图 2 中，点击红框处的选项后，将蓝框处的参数设为图 5-2 中的值，这个地址值是 SDRAM 的地址，为什么不填入一个 Flash 地址，而是以一个 SDRAM 地址来代替的原因是：在 BIOS 程序中，程序会自自把自己搬移到这个 SDRAM 地址处来执行。

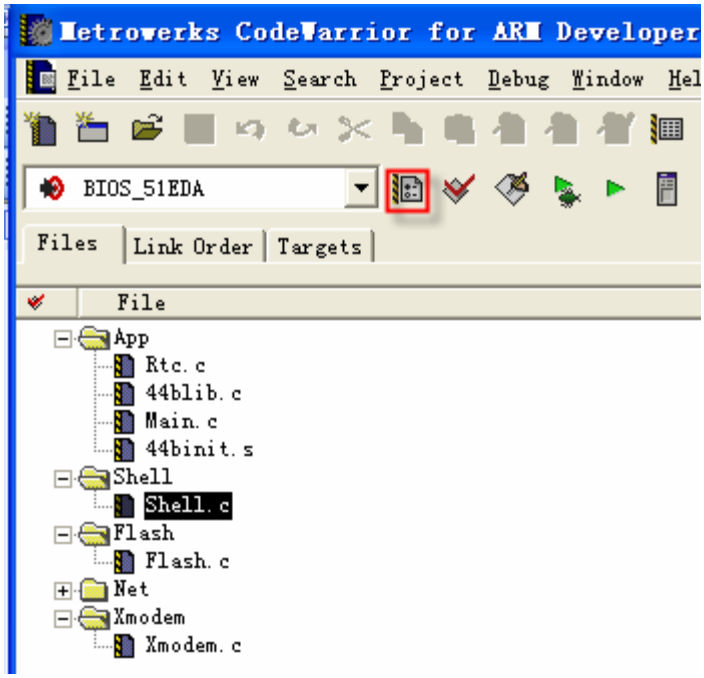


图　5-1

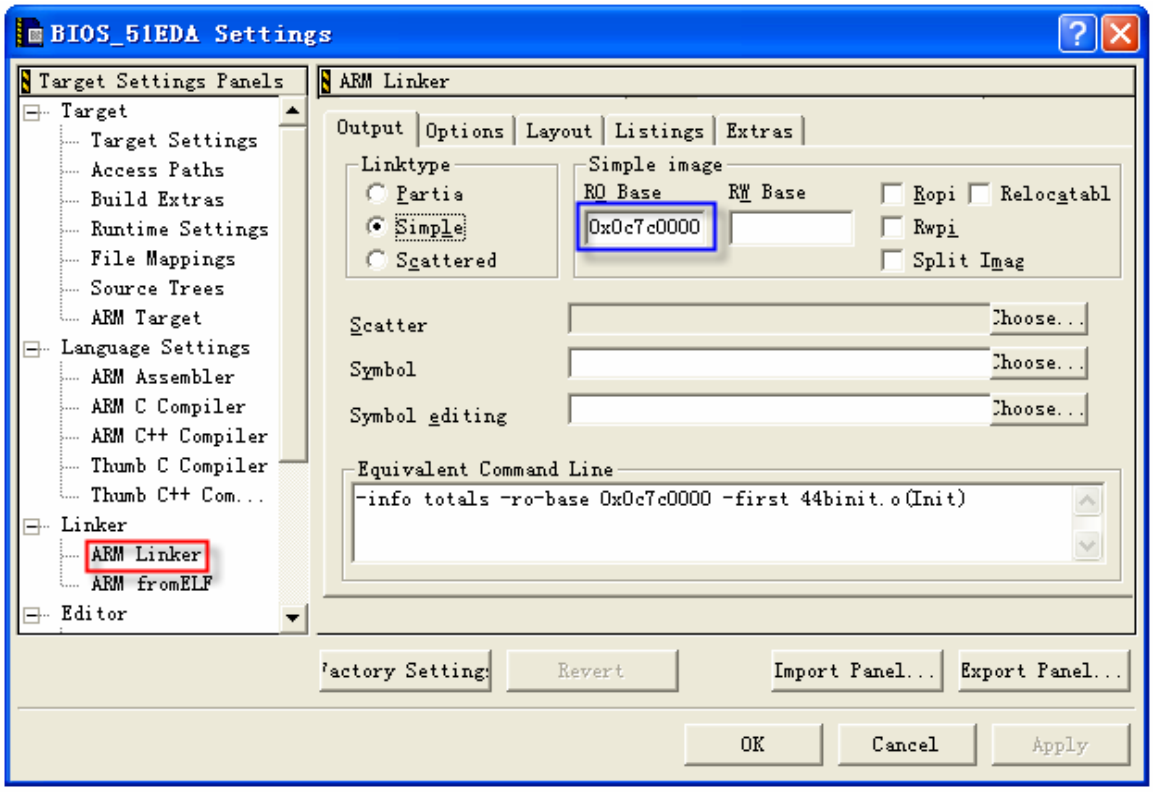


图　5-2

点击图 5-3 中红框所示的选项，在 Output format 选项中选择 ELF。

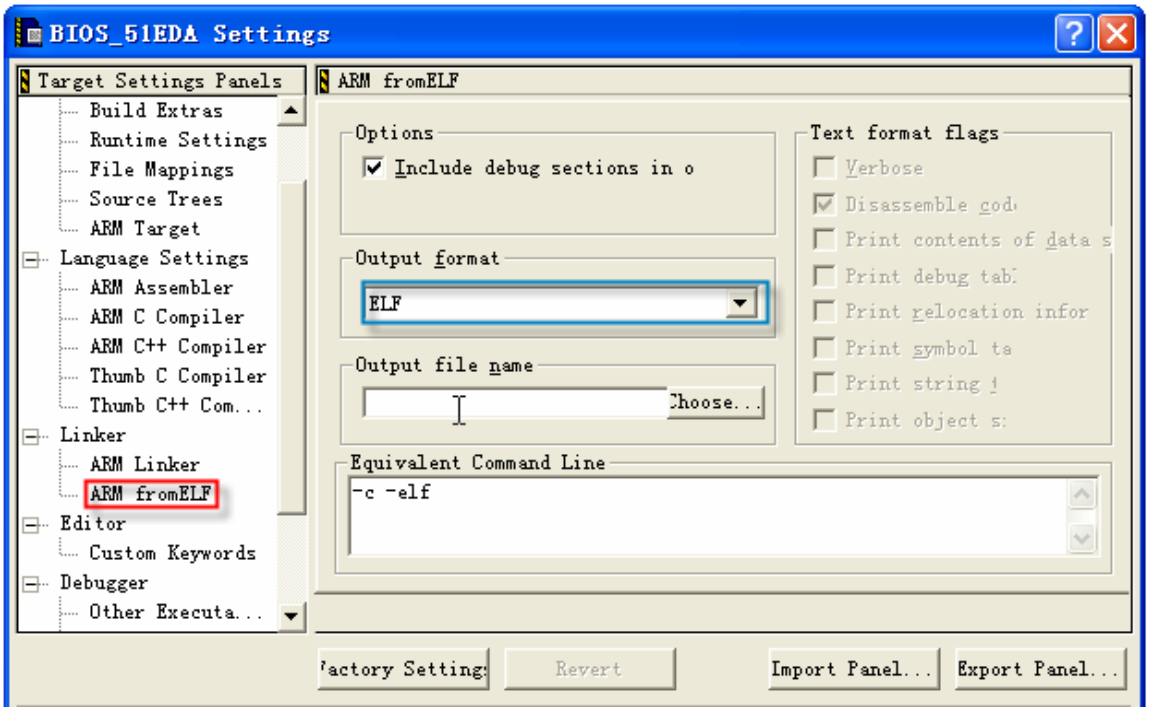


图　5-3

设置完成后，编译就可以在光盘中的Source\51EDA_QTBIOSV1.0\51EDA_BIOS_Data\BIOS_51EDA 文件夹里生成BIOS_51EDA.elf 文件，按第四章的步骤把这个文件烧写到开发板中就可以了。

3) BIOS 的初始波特率为 57600，默认串口是串口 1；

4) IP 地址为 192.168.1.18;

2、BIOS 的使用

1) 如何进入 BIOS 的命令行

开发板加电启动后，在超级终端中，我们可以看到以下的输出，系统会在 10 秒倒计时结束后，自动跳转到 0x10000 地址处执行，通常在这个地址上我们放置的是 uClinux 或是别的操作系统，也就是说会自动启动嵌入式操作系统。而当我们在 10 秒内在键盘上按任意时，就可以进入 BIOS 命令行，如图 5-5 所示。

```
*********************************
 Welcome to 51EDA BIOS Ver 1.0
 For QT44B0X ARM Develop Board
     http://www.51eda.com
*********************************

SDRAM Size: 16 MB
FLASH Type: SST39VF1601
IP Address: 192.168.1.18

Auto boot in progress, press any key in 7 seconds to run BIOS...
```

图 5-4

```
*********************************
 Welcome to 51EDA BIOS Ver 1.0
 For QT44B0X ARM Develop Board
     http://www.51eda.com
*********************************

SDRAM Size: 16 MB
FLASH Type: SST39VF1601
IP Address: 192.168.1.18

Auto boot in progress, press any key in 7 seconds to run BIOS...

\>
```

图 5-5

2) BIOS 功能说明

- **显示指令帮助信息**

在 BIOS 命令行下输入 help 指令，就可以看到如下图的提示信息

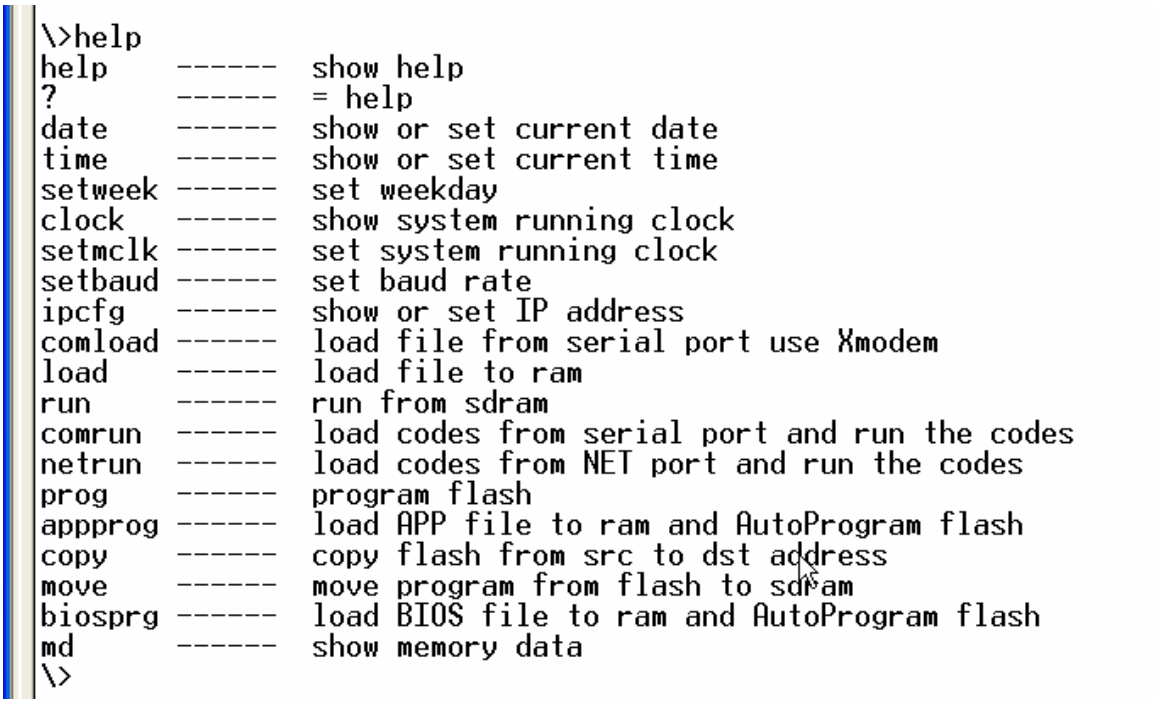


图 5-6

- **BIOS 指令简表**

BIOS 命令	命令提示信息	BIOS 命令说明
help	show help	显示 BIOS 命令列表
?	= help	显示BIOS 命令列表
date	show or set current date	显示或设置开发板时钟的当前日期
time	show or set current time	显示或设置开发板时钟的当前时间
setweek	set weekday	设置星期信息
clock	show system running clock	显示系统当前运行速度
setmclk	set system running clock	设置系统当前运行时钟
setbaud	set baud rate	设置串口通信波特率

ipcfg	show or set IP address	显示或设置当前开发板IP地址
comload	load file from serial port use Xmodem	通过串行口下载文件
load	load file to ram	通过网络下载文件
run	run from sdram	运行SDRAM中的程序
comrun	load codes from serial port and run the codes	通过串行口下载文件并运行该文件
netrun	load codes from NET port and run the codes	通过网络下载文件并运行该文件
prog	program flash	对FLASH存储器进行编程
appprog	load APP file to ram and AutoProgram flash	下载文件并编程写入FLASH存储器
copy	copy flash from src to dst address	从源地址到目标地址拷贝FLASH内容
move	move program from flash to sdram	从ADDR1 转移数据到ADDR2
biosprg	load BIOS file to ram and AutoProgram flash	通过网络下载 BIOS 文件，并编程写入 FLASH
md	show memory data	显示 SDRAM 中指定地址的数据

- **BIOS 指令具体说明**

所有指令的参数均为 16 进制，且输入时不需加 0x，如 load c100000，表示 load 指令的参数为 16 进制的 0xc100000。

help 和?

列出所有命令并给出简单的说明；

date

显示和设置当前日期，只输入 date 命令则显示日期，输入 date 2005-6-22-6-8 则设置当前日期为 2005 年 6 月 22 日；

time

显示和设置当前时间，只输入 time 命令则显示时间，输入 time 21:10:10 则设

置当前时间为 21 点 10 分 10 秒；

setweek

设置星期几，参数 n 从 1 到 7 表示星期一到星期日，比如 setweek 3，将开发板的的日期设置为周三；

clock

显示开发板当前的工作频率。如输入 clock，超级终端显示出当前的工作频率为 60MHz；

```
\>clock
System is running @ 60000000Hz
\>_
```

图 5-7

setmclk

该命令用于改变处理器的工作频率，具体参数设置可见芯片手册，注意不要使频率超出工作范围。执行该命令后，可以看到如下图的提示信息，输入几个参数后，就可以让处理器工作在别的频率上：

```
\>setmclk
This function is for adjust system running clock!!!
Current clock is 32000000Hz
Please enter the PLL parameter to use, mdiv[0-255], pdiv[0-63], sdiv[0-3]
mdiv: 64
pdiv: 10
sdiv: 0
You set System clock mdiv = 64, pdiv = 10, sdiv = 0
Current clock is 48000000Hz
```

图 5-8

setbaud

用于改变串口的波特率，下图显示了所支持的波特率，改完后要在 PC 上相应改变串口通讯波特率后再敲回车；

```
\>setbaud
0 : 4800
1 : 9600
2 : 19200
3 : 38400
4 : 57600
5 : 115200
Please enter choice :
```

图 5-9

ipcfg

显示和修改 tftp 下载时所用的 IP 地址，只输入 ipcfg 则显示当前 IP 地址，输入 ipcfg 192.168.1.100 则将 ip 地址改为 192.168.1.100；

comload

启动串口下载,若没带地址参数,则使用缺省下载地址 0x0c008000,若指定地址，下载数据保存到指定地址开始的 SDRAM 中去，如 comoad c100000，表明程序下载到 0xc100000 地址处；

load

启动 tftp 接收，若没带地址参数，则使用缺省下载地址 0x0c008000,若指定地址，下载数据保存到指定地址开始的 SDRAM 中去，如 load c300000。启动 tftp 接收后，要在 PC 端执行 tftp 程序，在 win2000 或 winxp 下，直接输入 tftp -i xxx.xxx.xxx.xxx put 文件名即可。如：

tftp -i 192.168.1.18 put image.rom

注意进行 tftp 传输时要保证 PC 机和开发板处于同一个 IP 段内；

run

运行存储器中的程序，缺省地址就是缺省下载地址 0xc008000，也可指定运行地址。如：

run 0xc100000

comrun

启动串口下载，并在接收完数据后自动运行下载的程序，缺省下载地址和指定参数同 comload；

netrun

通过网络下载程序，并在接收完数据后自动运行下载的程序，缺省下载地址和指定参数同 load；

prog

可以烧写 NOR FLASH，目前支持 SST39VF160。prog 命令完整的参数是：

prog addr1 addr2 length

其中 addr1 是要烧写的 FLASH 的地址，大于等于 0，小于 200000，字对齐，addr2 是 sdram 中要烧进 flash 的数据区起始地址，length 是要烧写的长度；

appprog

通过网络下载程序，并把它烧写到 NOR FLASH 中，默认的起始地址为 0x10000；

move

把存储器中 addr1 开始的长度为 size 的数据拷贝到 addr2 始的地址去，move 命令完整的参数是：

move addr1 addr2 length

其中 addr1 是要源地址，可以为 FLASH 的地址也可以是 SDRAM 地址，字对齐，addr2 是 SDRAM 中目标数据区起始地址，length 是要烧写的长度；

biosprg

通过网络下载 BIOS 文件，并把它烧写到 NOR FLASH 的 0 地址处，覆盖原开发板上的 BIOS，请谨慎使用；

md

显示存储器中的指定地址开始的数据，总共显示 352 字节的内容，如下图所示：

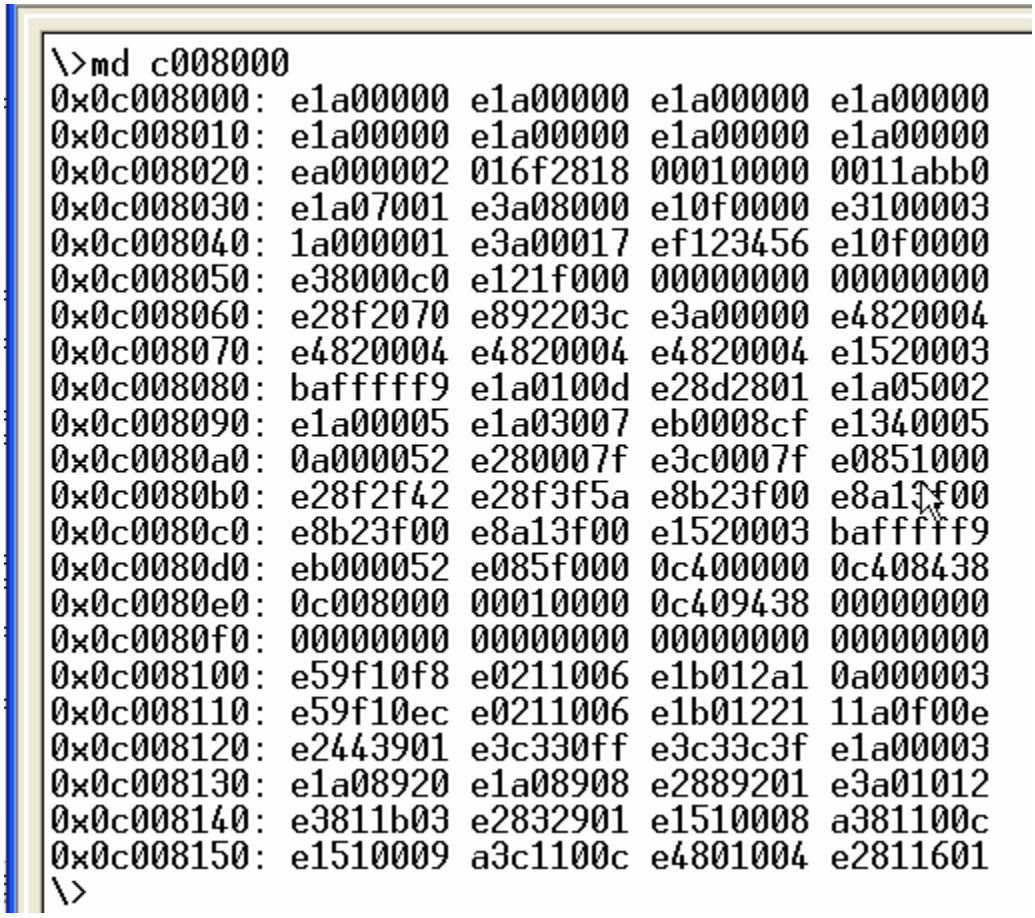


图 5-10

3、如何用 BIOS 烧写 uClinux 的映像文件

1、准备 uClinux 的映像文件

在配套光盘的“烧写文件”文件夹中，我们提供两个已经编译好的 uClinux 映像文件，image.rom 和 image.ram，前者为烧写到 FLASH 中的映像文件，后者则是可以下载到 SDRAM 中，直接执行的映像。为了方便使用，建议用户把这两个文件拷贝到硬盘某一分区的根目录下，比如 E:\。

2、进入 BIOS 命令行

按上一小节的方法，进入 BIOS 的命令行后，执行指令

approg

提示信息如下图所示：

```
*********************************
 Welcome to 51EDA BIOS Ver 1.0
 For QT44B0X ARM Develop Board
     http://www.51eda.com
*********************************

SDRAM Size: 16 MB
FLASH Type: SST39VF1601
IP Address: 192.168.1.18

Auto boot in progress, press any key in 2 seconds to run BIOS...

\>appprog
Now begin address for download, use address 0xc008000
Load image file from host
Type tftp -i 192.168.1.18 put filename at the host PC
Press ESC key to exit
```

图　5-11

这时，开发板已经做好准备从 PC 机上下载映像文件。

3、下载并烧写映像文件

在 PC 机上执行命令

tftp -I 192.168.1.18 put image.rom

传送完毕时会出现如下提示信息。

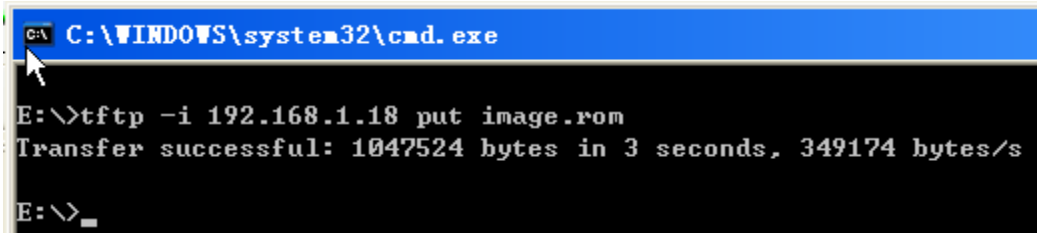


图　5-12

这时，在超级终端上可以看到如下提示信息，询问用户是否需要把下载的文件烧写到 FLASH 中，当我们键入”y”时，uClinux 的映像文件就被烧写到 FLASH 中了。

```
\>appprog
Now begin address for download, use address 0xc008000
Load image file from host
Type tftp -i 192.168.1.18 put filename at the host PC
Press ESC key to exit
Begin to receive filename: image.rom, transtype: octet.
TFTP transfer finished, and receive ffbe4 bytes.
program flash begin @0x10000, from ram data @0xc008000, size = 1047524Bytes
Are you sure? [y/n]
```

图　5-13

4、如何启动 uClinux

复位开发板，在 BIOS 进行倒数时，不要按键，10 秒倒计时结束后，BIOS 会自动启动 uClinux，下图为 uClinux 启动后的 LOGO。

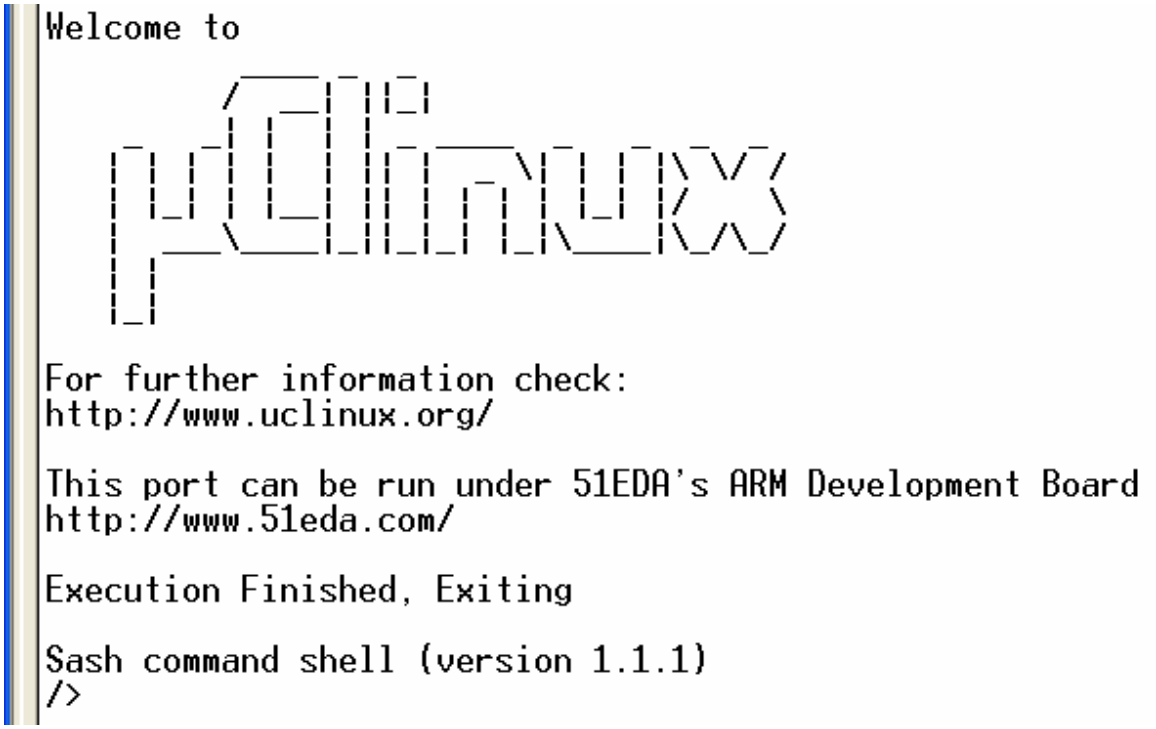


图 5-14

5、在 SDRAM 中运行 uClinux

在我们在调试 uClinux 的内核时，并不需要每次都把 uClinux 映像文件烧写到 FLASH 中，这时我们可以把映像文件下载到 SDRAM 中，并在 SDRAM 中执行。

在 BIOS 命令行中，输入指令

netrun

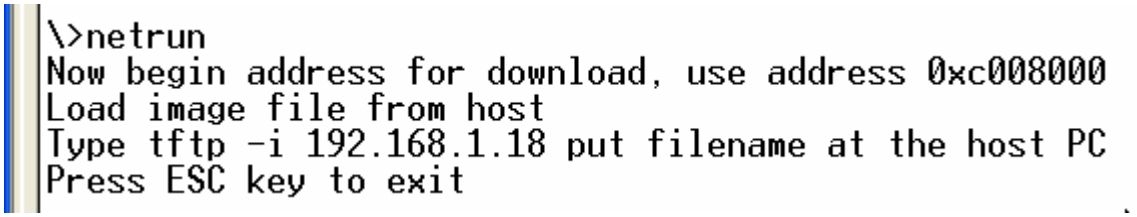


图 5-15

然后在 PC 机上输入命令

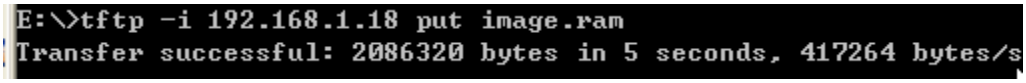


图 5-16

这时表明程序已经下载完毕，在超级终端中，我们可以看到当程序下载完成后，BIOS 会自动从 SDRAM 中执行刚才下载的程序，即启动 uClinux。

第六章　在 QT44B0X-I 上使用液晶

一、LCD 控制器

一般的液晶显示控制器主要由接口部分，驱动部分和控制部分组成。液晶显示控制器的接口部分由指令寄存器 / 译码器、数据缓冲器和状态字寄存器以及相关逻辑电路组成。它用来接收处理器发来的指令和数据，并向处理器反馈所需的数据信息。驱动部分是液晶显示控制器对液晶显示驱动系统的接口。时序发生器产生基础时钟提供给显示时序电路，显示时序电路产生显示时序脉冲序列提供给驱动部分。液晶显示控制器的控制部分是液晶显示控制器的核心。产生工作的时钟直接提供给时序发生器以生成控制时序和显示时序。控制时序将驱动逻辑电路以管理和操作各功能电路。

S3C44B0X 内置 LCD Controller 功能是非常强大的，可以用来控制 2，4，16 级灰度的单色 STN LCD 以及 256 色的彩色 DSTN LCD。并且 LCD 的刷新率可通过设置 CPU 内部的寄存器来进行调节。S3C44B0X 的 LCD 控制器主要功能就是把在系统内存的视频数据传送到外部的 LCD 驱动器。S3C44B0X 的 LCD 控制器由以下各功能模块组成，如下图所示：

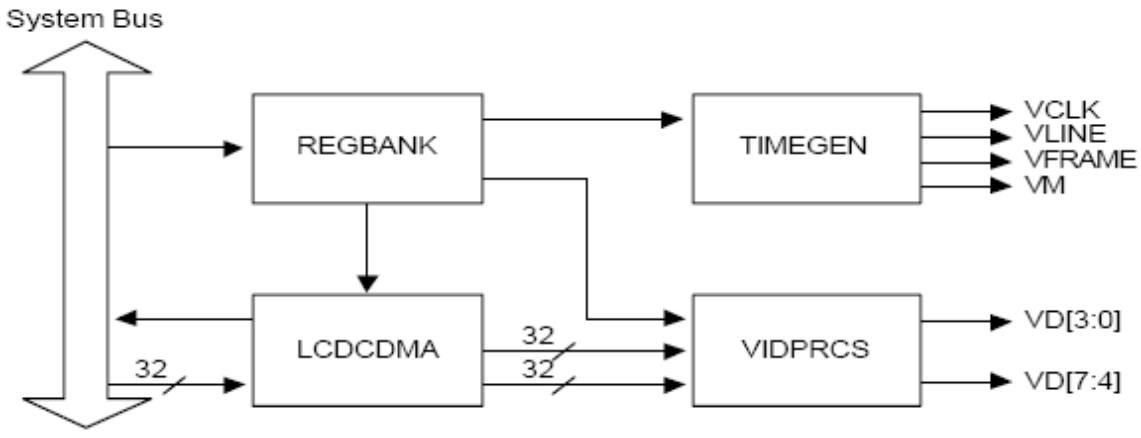


图　6-1

下面对所涉及到的模块的功能做简单的介绍：

- REGBANK 模块：对 LCD 控制器的配置

- LCDCDMA 模块：自动的把帧缓冲中的视频数据传到 LCD 驱动器
- VIDPRCS 模块：把从 LCDCDMA 模块传过来的数据经过适当的格式转换后通过 VD[7:0]口发送到 LCD 驱动器。
- TIMEGEN 模块：由可编程逻辑组成，支持多种不同的 LCD 驱动器的时序和频率。这个模块产生 VFRAME，VLINE，VCLK，VM 等信号。

特殊寄存器

S3C44BOX 的 LCD 控制器中主要含有下列特殊寄存器：

- LCDCONI：主要负责各模式设定，时钟寄存器等等；
- LCDCON2：主要负责垂直，水平扫描参数的设定；
- LCDCON3：测试模式的使能设定；
- LCDSADDRI：帧缓冲开始地址 1：
- LCDSADDR2：帧缓冲开始地址 2;
- LCDSADDR3：虚拟屏幕地址设置；
- REDLUT, GREENLUT，BLUELUT：红、绿、蓝三原色查找表寄存器。

以上这些寄存器通常在 LCD 初化时进行设置，具体的值与您所使用的 LCD 有关，具体设置的值请参考配套光盘中测试代码 LCD 部分的初始化代码。

二、如何在 ADS 环境中在 S3C44B0X 上面显示图片

1、将图片取模；

对于 S3C44B0X 来说，图片转换取模通常都用 uC/GUI 所带的图片转换工具软件——uC-GUI-BitmapConvert.exe，这是一个不错的软件。当然也有其他的工具软件，用法都类似。该软件在配套光盘中已有收录。

在将图片取模之前，通常要对图片进行处理，这大都使用 photoshop、ACDsee 甚至 Windows 系统自带的画图来进行。注意 uC-GUI-BitmapConvert.exe 只能识别 bmp 格式的图片，所以其他格式的图片要首先转换成 bmp 格式的图片，图片的尺寸可以根据需要，使用这些软件来裁剪、处理。

首先运行 uC-GUI-BitmapConvert.exe，再点击菜单 File -> open，这时会弹出

一个浏览的窗口；在浏览的窗口里面找到你已经处理的图片并打开，现在你就可以看到菜单下方的区域出现了你刚打开的图片了。

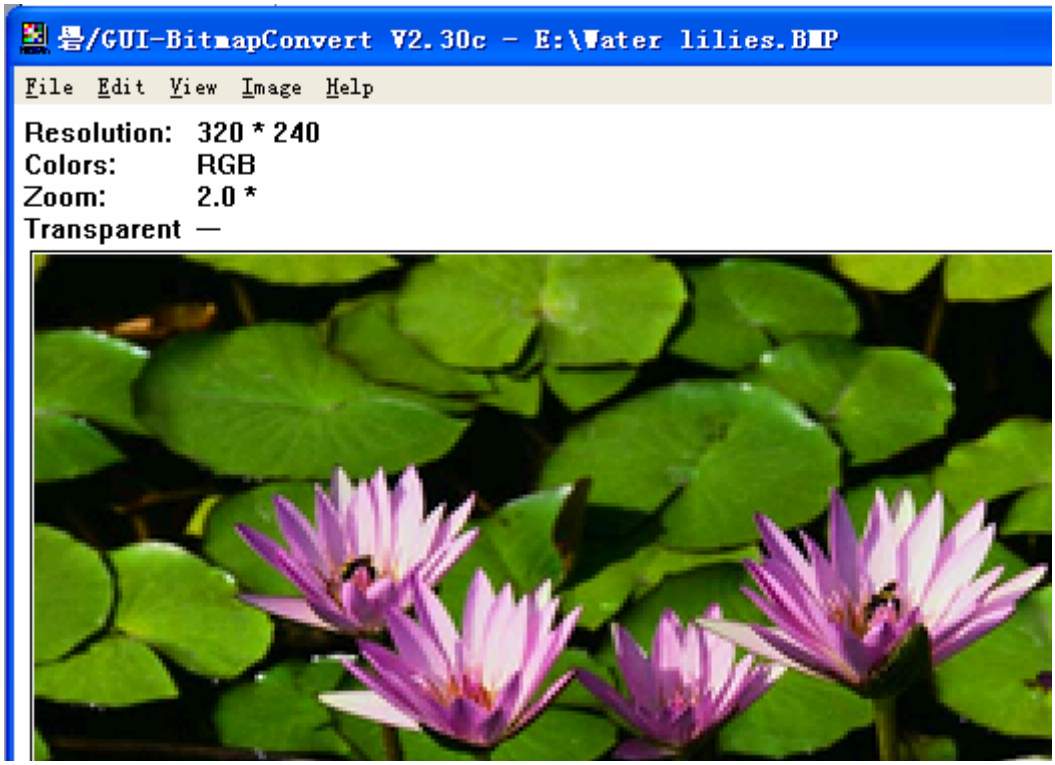


图 6-2

- 从图片中取出黑白效果的模；

现在点击菜单 Image -> Convert into -> BW；这时就可以看到你刚才打开的图片变成黑白效果了。接着点击菜单 File -> Save as，这时会弹出一个浏览的窗口；在浏览的窗口里面你可以选择取模文件的存放路径，也可以指定文件名，文件保存类型使用默认类型——“C” with palette 就好了。

- 从图片中取出 4 级灰度效果的模；

现在点击菜单 Image -> Convert into -> Gray4；这时就可以看到你刚才打开的图片变成 4 级灰度效果了。接着点击菜单 File -> Save as，这时会弹出一个浏览的窗口；在浏览的窗口里面你可以选择取模文件的存放路径，也可以指定文件名，文件保存类型使用默认类型——“C” with palette 就好了。

- 从图片中取出 16 级灰度效果的模；

现在点击菜单 Image -> Convert into -> Gray16；这时就可以看到你刚才打开

的图片变成 16 级灰度效果了。接着点击菜单 File -> Save as，这时会弹出一个浏览的窗口；在浏览的窗口里面你可以选择取模文件的存放路径，也可以指定文件名，文件保存类型使用默认类型——“C” with palette 就好了。

- 从图片中取出 256 伪彩色效果的模；

现在点击菜单 Image -> Convert into -> 8bit color 233；这时就可以看到你刚才打开的图片变成 256 伪彩色效果了。接着点击菜单 File -> Save as，这时会弹出一个浏览的窗口；在浏览的窗口里面你可以选择取模文件的存放路径，也可以指定文件名，文件保存类型使用默认类型——“C” with palette 就好了。

2、字模文件处理；

现在需要对刚才取模的文件进行一些小小的处理了，就拿一个例子的示范吧。

譬如打开的一个图片名字叫 256LCD.bmp，图片尺寸为 240×320 象素。采用 256 伪彩色取模，取模后保存的文件叫做 256LCD.c。

现在需要打开这个 C 文件，删除“unsigned char ac256LCD[]”之前的全部内容，并删除“const GUI_BITMAP bm256LCD”和它以后的全部内容，整个文件里面仅保留一个如下的数组，不需要再有其他多余的东西:

```
unsigned char ac256LCD[] = {
0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27,
0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27,
……
0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27,
0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27
};
```

3、图片显示；

现在所要做的工作就是把这个 C 文件添加到你的 ADS1.20 项目里面去，并在需要显示图片的源码文件的前面或者头文件中添加一条：

```
extern unsigned char ac256LCD[] ;（示例，其他取模文件也类似）
```

然后你就可以调用该图片显示函数来显示该图片了，这个图片是 256 色伪彩色的，尺寸是 240×320，所以要调用：

void Lcd_C256_Bmp(U16 x0, U16 y0, U16 x1, U16 y1 , unsigned char bmp[])

这里面的（x0，y0）是图片起始点的坐标，（x1，y1）是图片结束点的坐标。

注意（x1－x0）必须要等于图片的水平象素，（y1－y0）要等于图片的垂直象素。

bmp[]是图片数组的入口；

具体的调用过程如下：

Lcd_C256_Bmp(0，0，240，320，ac256LCD)；

OK，现在编译项目，下载或烧写到目标板运行，就可以看到图片了。

图 6-3

对于黑白图片、4 级灰度图片、16 级灰度图片，过程也类似，这里不多说了。

三、FrameBuffer 介绍

从这一小节开始，我们涉及了 uClinux 和设备驱动的一些编程知识，如果对以下小节的内容不太明白，可以先读完后面章节后再回过头来重读这几小节。

编写 uClinux 下的 LCD 驱动，就会涉及到 FrameBuffer。FrameBuffer,帧缓冲，有时简称为 fbdrv,基于 fbdrv 的 console 也被称之为 fbcon。这是一种独立于硬件的抽象图形设备。帧缓冲设备对图象硬件设备提供了一种抽象化处理. 它代表了一些视频硬件设备，允许应用软件通过定义明确的界面来访问图象硬件设备，因此软件无需了解任何涉及硬件底层驱动的东西（如硬件寄存器）。FrameBuffer 的优点在于其高度的可移植性、易使用性、稳定性。帧缓冲设备属于字符设备，采用“文件层----驱动层”的接口方式. 在文件层次上，Linux 为其定义了 file_operations 的数据结构：

```
static struct file_operations fb_fops = {
  owner: THIS MODULE,
  read: fb_read, /*读操作*/
  write: fb_write, /*写操作*/
  ioctl: fb_ioctl, /*控制操作*/
  mmap: fb_map, /*映射操作*/
  open: fb_open, /*打开操作*/
  release: fb_release, /*释放操作*/
 };
```

其中的成员函数都在文件 linux-2.4.x/driver/video/fbmem. c 中定义。由于显示设备的特殊性，在驱动层的接口中不但要包含底层函数，还要有一些记录设备状态的数据。Linux 为帧缓冲设备定义的驱动层接口为 struct fb_info 结构，在 include/linux/fb.h 中定义。幸运的是，嵌入式系统要求的显示操作比较简单，只涉及到结构中少数几个成员。

四、LCD 的驱动程序

LCD 在 uClinux 下的驱动实际上与帧缓冲的内容相关，主要涉及到以下一些文件、数据结构和函数：

主要文件

- fbcon.c：通用的基于 framebuffer 的 console 的实现；
- fbcon-cfb16.c：底层的基于 framebuffer 的 console 的 16bpp 驱动的实现；
- fbmem.c：对 framebuffer 的通用操作接口；
- fbcmap.c：处理 color map;
- modedb.c：各显示 mode 的数据库以及操作；
- fonts.c：该文件处理字体；
- s3c44b0xfb.c：针对具体的 LCD 和液晶控制器的驱动程序，这个文件需要我们根据具体的 LCD 进行修改或添加。

在帧缓冲设备控制台(console)驱动中，主要由三部分组成：一是 fbdev，控制了具体的硬件设备，二是 fbcon，是通用的帧缓冲设备控制台，三是 fbcon-*,底层的绘制程序的实现。

数据结构

LCD 驱动主要的数据结构有：fb_info，fb_fix_screeninfo，fb_var_screeninfo 以及针对特定的 LCD 控制器的数据结构，这里是 s3c44b0fb info。

fb_info 中纪录了帧缓冲设备的全部信息，包括设备的设置参数，状态以及操作函数指针。每一个帧缓冲设备都必须对应一个 fb_info 结构。其中成员变量 modename 为设备名称，fontname 为显示字体，fbops 为指向底层操作的函数的指针，这些函数是需要驱动程序开发人员编写的。成员 fb_var_screeninfo 和 fb_fix_screeninfo 也是结构体。其中 fb_var_screeninfo 记录用户可修改的显示控制器参数，包括屏幕分辨率和每个像素点的比特数。fb_var_screeninfo 中的 xres 定义屏幕一行有多少个点，yres 定义屏幕一列有多少个点，bits_per pixel 定义每个点用多少个字节表示。它的信息可以通过 ioctl 函数的 FBIOGET_VSCREENINFO 获得，可以通过 FBIOPUT_VSCREENINFO 更新。fb_fix_screeninfo 中记录用户不能修改的显示控制器的参数，如屏幕缓冲区的物理地址，长度。同样，它的信息可以通过 ioctl 的 FBIOGET_FSCREENINFO 获得。当对帧缓冲设备进行映射操作的时候，就是从 fb_fix_screeninf。中取得缓冲区物理地址的。上面所说的数据成员都是需要在驱动程序中初始化和设置的。

结构体 fb_var_screeninfo 中一些主要成员如下：

```
    struct fb_var_screeninfo {
        _u32   xres, yres, xres_virtual, yres_virtul;
        _u32   xoffset, yoffset;
        _u32   bits_per_pixel;
        struct fb_bitfield red, green, blue, transp;
        _u32   pixclock;
        _u32   left_margin, right_margin, upper_margin, lower_margin;
        _u32   hsync_len, vsync_len;
};
```

主要函数

在 uClinux-dist/linux-2.4.x/drivers/video/s3c44b0xfb.c 文件中，其主要操作为：

```
static struct fb_ops s3c44b0fb_ops＝ {
       owner：THIS MODULE,
       fb_get_fix：s3c44b0fb_get_fix,
       fb_get_var：s3c44b0fb_get_var,
       fb_set_var: s3c44b0fb_set_var,
      fb_get_cmap: s3c44b0fb_get_cmap,
      fb_set_cmap：s3c44b0fb_set_cmap,
};
```

- s3c44b0fb_init

初始化函数：初始化函数首先初始化 LCD 控制器，通过写寄存器设置显示模式和显示颜色数，然后分配 LCD 显示缓冲区。在 uCLinux 可通过 kmalloc 函数分配一片连续的空间。缓冲区通常分配在大容量的片外 SDRAM 中，起始地址保存在 LCD 控制器寄存器中。最后是初始化一个 fb_info 结构，填充其中的成员变量，并调用 register_framebuffer(&fb_info)，将 fb_info 在内核中登记。

以下是 init 函数的简要分析：

```
    int __init s3c4460fb_init(void)
     {
        struct s3c44b0fb_info *fbi：
        int ret ＝ -ENOMEM;
        fbi = s3c44b0fb_init_fbinfo();
        //开辟 fbi 的数据结构的空间，初始化，设置 fb.fix，fb.var（res，bpp 等等）的参数。
        if (!fbi)
              goto failed：
```

```
          ret = s3c44b0fb_map_video_memory(fbi);
          //开辟 frame buffer 的内存空间,初始地址为 fbi->fb.fix.smem _start,
大小为 fbi->fb.fix.smem_len
          if  (ret)
              goto failed;
          s3c44b0_lcd_init();
          //LCD 控制端口初始化
          s3c44b0fb_set_var(&fbi->fb.var, -1, &fbi->fb);
          ret = register_framebuffer(&fbi->fb):
          MOD_INC_USE_COUNT
          return 0;
failed:
          if (fbi)
                kfree(fbi):
           return ret;
       }
```

fb_ops 结构中的函数都是用来设置或者获取 fb_info 结构中的成员变量的。当应用程序对设备文件进行 ioctl 操作时候会调用它们。

- s3c4460fb_get_fix

应用程序传入的是 fb_fix_screeninfo 结构，在函数中对其成员变量赋值，主要是对固定值 smem_start（缓冲区起始地址）和 smem_len（缓冲区长度）赋值，并最终返回给应用程序。

```
       static int s3c4460fb_get_fix(struct fb_fix_screeninfo *fix,
            int con, struct fb_info *info)
       {
           struct display *display ＝get_con_display(info, con);
           *fix = info->fix
           fix->line_length = display->line_length:
           fix->visual = display->visual;
           return 0;
       }
```

- s3c4460fb_get_var

这个函数主要是得到一些可变的参数值，如实际显示区域 xres、yres，虚拟显示区域 xres virtual、yres_virtual 等.

```
       static int s3c44b0fb_get_var(struct fbvar_screeninfo *var, int con,
            struct fb_info *info)
       {
```

```
            *var = *get_con_var (info, con);
            return 0;
        }
```

- s3c44b0fb_set_var

s3c44b0fb_set_var()函数的传入参数是 fb_var_screeninfo，函数中需要 对 xres，yres,和 bits_per_pixel 赋值。

由于 s3c44b0xfb.c 文件与用户将使用的液晶有很大的关系,在此不给出源代码，请大家在深入学习后，自行完成相关的驱动程序。

第七章　QT44B0X-I 如何恢复到出厂设置

QT44B0X-I 开发板出厂之前 BIOS 已经写好，如果在您的使用过程中不慎冲掉，可以按以下步骤重新烧写：

1、连接 PC 和开发板

把三合一 JTAG 板插到 PC 的并口上，并要在 BIOS 中把相应并口的工作模式设为 EPP，将 20 芯排线将开发板的 JTAG 接口和三合一 JTAG 板的 Wiggler 接口相连，并打开电源；

2、烧写 BIOS

在 Flashpgm 里点击 program，如下图所示；

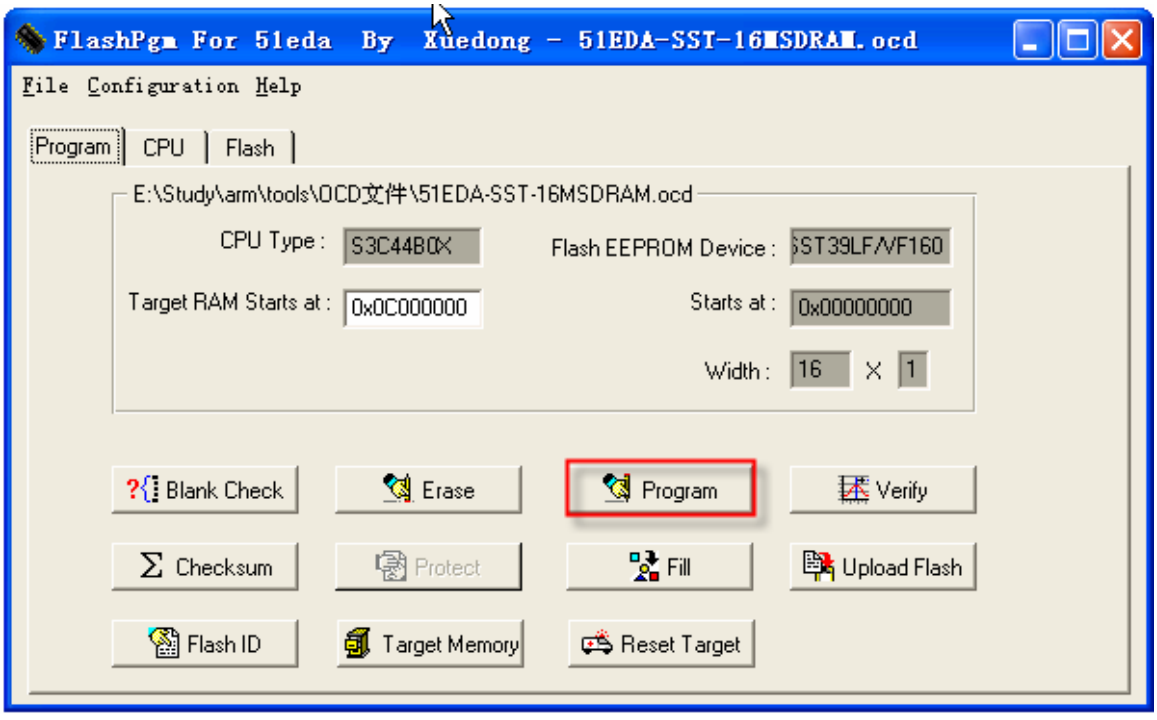


图　7-1

点击 Browse 按钮，在光盘中找到“烧写文件”文件夹下的 51EDA_BIOS.elf 文件；

点击 Erase Target Flash Sector(s.) Before Pro 前面的小窗口，使其前面打

勾；

点击 Program 按键开始烧写，10 秒钟左右即可烧完；

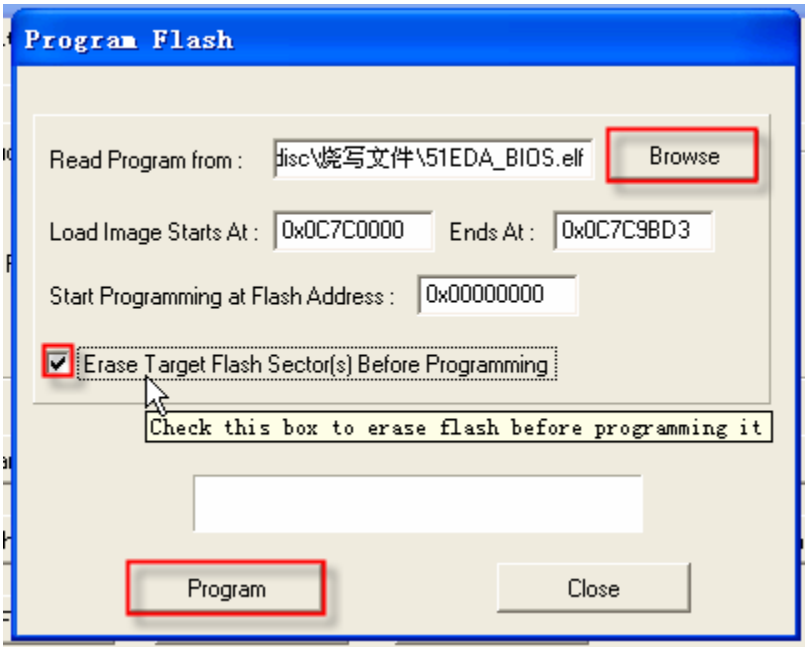


图 7-2

然后点击 Close 按键关闭烧写界面，在超级终端中可以看到 BIOS 启动后的输出信息，超级终端的设置就参看第二章的内容。

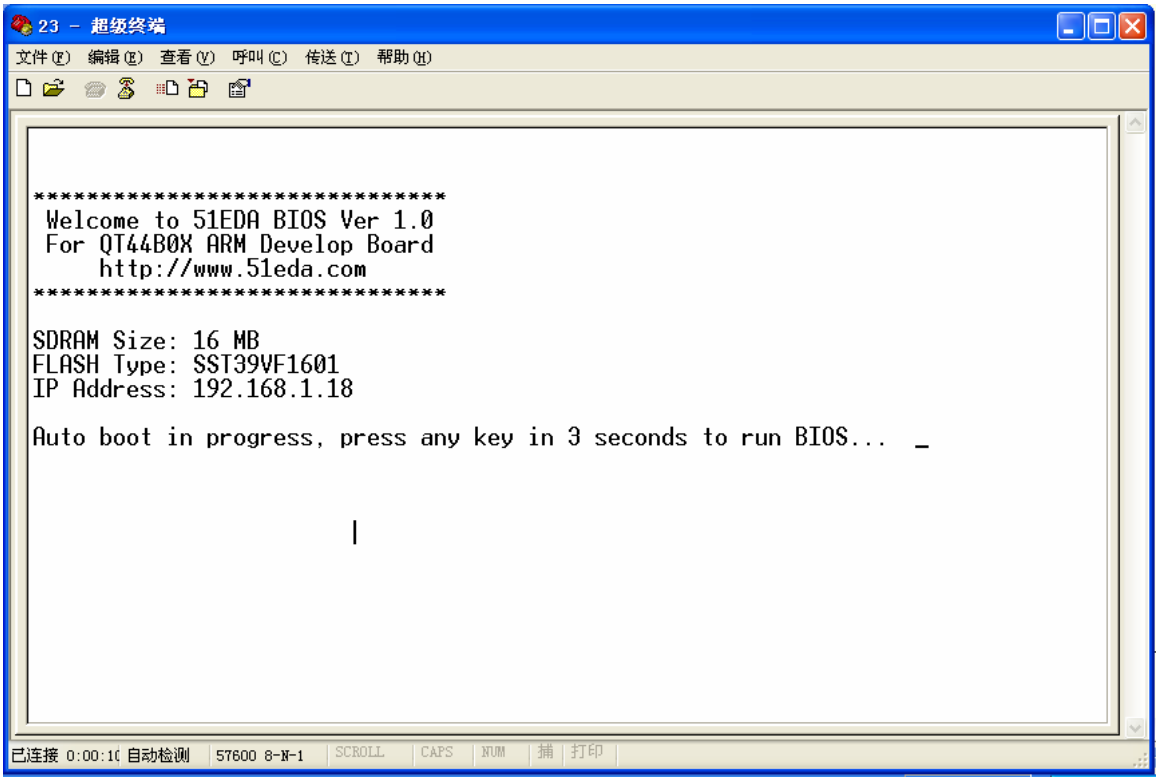

23 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
****
Welcome to 51EDA BIOS Ver 1.0
For QT44B0X ARM Develop Board
http://www.51eda.com
****
SDRAM Size: 16 MB
FLASH Type: SST39VF1601
IP Address: 192.168.1.18
Auto boot in progress, press any key in 3 seconds to run BIOS...
已连接 0:00:1(自动检测
57600 8-N-1
SCROLL
CAPS
NUM
捕
打印

图 7-3

第八章　QT44B0X-I 开发板使用 FAQ

1、如何重新烧写 BIOS?

答：请参看下一章的“QT44B0X-I 如何恢复到出厂设置”

2、为何超级终端不能输入？

答：将超级终端 COM1 或 COM2 属性/端口设置中的流量控制设置为---无

3、为何超级终端里，BIOS 不接受 HELP 等指令？

答：BIOS 的指令都是小写，只能输入 HELP 等指令

4、为何网络 ping 不通？

答：请将 PC 机 IP 地址设置为 192.168.1.XXX（X 为 1-254 的数字，但注意不要和开发板的 IP 相冲突），子网掩码设置为 255.255.255.0，QT44B0X-I 的 BIOS 启动后,在超级终端里输入 load 并回车。再在 windows 里运行 ping 192.168.1.8 -t，即可 ping 通。

5、为何 3 合 1 的 JTAG 连不上目标板？

答：这个一般是因为连线没有连好，或者计算机 BIOS 里面没有把并口设置成 EPP 模式，或者是设置不正确。发现有些笔记本无法使用 JTAG，但台式机都没有问题，建议到笔记本官方网站或者网上查找一下相关的 EPP 模式的设置方法或换成台式机来调试。

6、FlashPGM 无法进行 Flash 烧写怎么办？

答：首先请确认你使用的 FlashPGM 已经解密了，正确设置通讯电缆类型、并口端口、OCD 配置文件等。请参阅 FlashPGM 如何使用，观看我们的多媒体演示文件。

7.超级终端输出的怎么是乱码?

答：这个由于波特率设置不正确，遇到这种情况，换个波特率看看，一定能解决。

第二部分　uClinux

第九章　嵌入系操作系统uClinux的简述

操作系统是管理计算机上的资源,为用户使用计算机及其外部设备提供最基本接口的程序。

自从计算机诞生以来，随着计算机、网络技术的快速发展，操作系统一直处于不断发展和改进之中,人们将越来越多的功能加入到操作系统中,导致操作系统越来越大。但是，随着应用领域的扩大，为了适应不同的应用场合，考虑到系统的灵活性、可伸缩性以及可裁减性，一种以应用为中心、以计算机技术为基础、软件硬件可裁剪、适应应用系统对功能、可靠性、成本、体积、功耗要求严格的专用计算机系统---嵌入式操作系统便随之延生。

uClinux 是嵌入式 Linux 的一个分支，所以我们首先来了解一下嵌入式 Linux 的特性。嵌入式 Linux 是按照上面所说的嵌入式操作系统的要求设计的一种小型操作系统。由一个 kernel（内核）及一些根据需要进行定制的系统模块组成。其 Kernel 很小，一般只有几百 K 左右，即使加上其它必要的模块和应用程序，所需的存储空间也很小，有些还有具有实时性，如 Rtlinux。

一个小型的嵌入式 Linux 系统只需要下面三个基本元素：

- 引导程序
- Linux 微内核（由内存管理、进程管理和事务处理构成）
- 初始化进程

如果要让它有更多的功能且继续保持小型化，可以加上：

- 文件系统
- TCP / IP 网络支持
- 存储更多数据用的磁盘。
- 提供设计精简的应用程序。

运行嵌入式Linux的CPU可以是 X86,Alpha,Sparc,MIPS,PPC,MOTOROLA,NEC,

ARM。与这些芯片搭配的主板都很小，与一张 PCI 卡大小相当，有的甚至更小！

嵌入式 Linux 所需的存储器不是软盘、硬盘、ZIP 盘、CD-ROM、DVD 这些众所周知的常规存储器，它使用 Rom，CompactFlash 等体积较小，存储容量不太大的存储器，在 51EDA 的开发板上采用的是 Nand Flash。它的“内存”可以使用普通的内存也可以使用专用的 RAM。

PC 平台上通常有键盘、鼠标、显卡、显示器等输入输出设备及各种存储设备，有 ISA、PCI、AGP、USB 等接口。在嵌入式系统中，往往不需要上述设备和接口的支持，而使用嵌入式系统特有的接口和设备，如 LCD 显示屏、触摸屏,I2C，I2S 等,此外 51EDA 的开发板上还提供了 IDE、PS2、USB 等丰富的接口。在一个简单的系统里，当系统启动后，内核和所有的应用程序都在内存里。这就是大多数传统的嵌入式系统工作模式，它可以被 Linux 支持。因此嵌入式系统可以没有磁盘。许多嵌入式系统没有磁盘或者文件系统。Linux 不需要它们也能运行。有多种途径可以消除对磁盘的依赖，这要看系统的复杂性和硬件的设计。

在 uClinux 这个英文单词中 u 表示 Micro，微的意思，C 表示 Control，控制的意思,所以 uClinux 就是 Micro-Control-Linux,即“面向微控制器的 Linux 系统”。

1、uClinux 的内核加载方式

uClinux 的内核有两种可选的运行方式：可以在 flash 上直接运行，也可以加载到内存中运行。

Flash 运行方式：把内核的可执行映象文件烧写到 flash 上，系统启动时从 flash 的某个地址开始逐句执行。这种方法实际上是很多嵌入式系统采用的方法。这种做法可以减少内存需要。

内核加载方式：把内核的压缩文件存放在 flash 上，系统启动时读取压缩文件在内存里解压,然后开始执行,这种方式相对复杂一些,但是运行速度可能更快(RAM 的存取速率要比 flash 高）。同时这也是标准 Linux 系统采用的启动方式。

2、uClinux 的根（root）文件系统

uClinux 系统采用 romfs 文件系统，这是一种轻量级的文件系统，其相对于一

般的 ext2 文件系统要求更少的空间。空间的节约来自于两个方面，首先内核支持 romfs 文件系统比支持 ext2 文件系统需要更少的代码，其次 romfs 文件系统相对简单，在建立文件系统超级块（superblock）需要更少的存储空间。romfs 文件系统不支持动态擦写保存，对于系统需要动态保存的数据采用虚拟 ram 盘的方法进行处理，而 ram 盘将采用 ext2 文件系统。

3、uClinux 的内存管理方式

uClinux 主要面向的是没有 MMU(Memory Management Unit)处理器的嵌入式系统，因此它的内存管理方式与我们 PC 上的 linux 系统有较大的差异，我们将在下文中加以详述。

第十章　开发模式和交叉编译环境的建立

1、主机和目标板的开发模式

目前绝大多数的嵌入式系统都是采用如下图中的主机－目标板的开发模式。

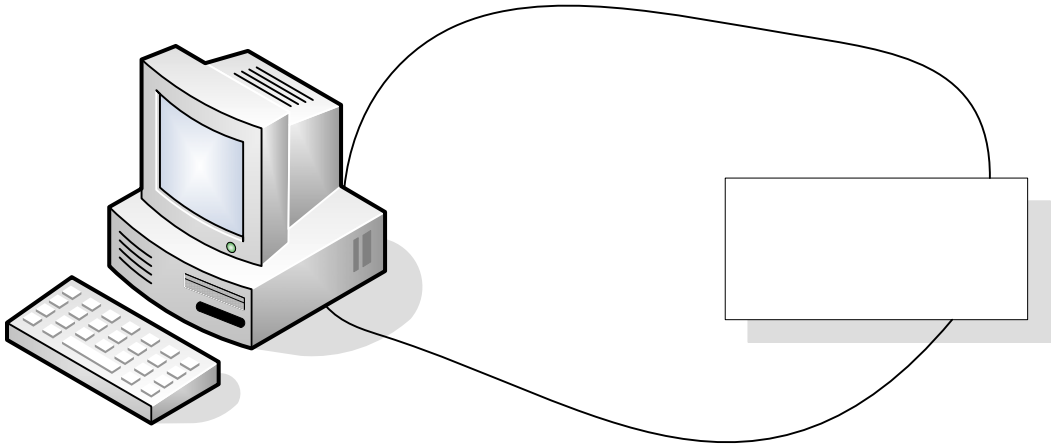


图　10-1

其中主机上推荐安装 Linux 操作系统，为下面的交叉编译提供方便的操作平台。

2、交叉编译环境

我们在普通软件的开发过程中，通常是在本机开发、调试，并在本机上运行的。这种方式通常不适合于嵌入式系统的软件开发，因为对于嵌入式系统的开发，没有足够的资源在ARM目标板上运行开发工具和调试工具。比如说，没有足够的存储空间和足够快速的CPU。所以，通常的嵌入式系统的软件开发采用一种交叉编译调试的方式。所谓交叉编译，简单地说，就是在一个平台上生成另一个平台上的可执行代码。这里需要注意的是所谓平台，实际上包含两个概念：体系结构（Architecture）、操作系统（Operating System）。在我们这个开发模式下，交叉编译是指在x86（PC机）的体系结构下编译生成ARM体系结构的代码。

交叉编译环境的建立是一个复杂且烦琐的过程，在51EDA的开发板上，我们提供

了一套已经设置好的交叉编译环境，您所需要做的就是将它安装到您的主机上。这个安装文件位于51EDA配套光盘的Tools\uclinux_tools目录下。安装时需要管理员权限，且Linux下的所有指令都区分大小写。

ls /mnt　　列出/mnt目录下的内容，看是否已经存在cdrom这个目录
　　若没有cdrom这个目录，则

mkdir /mnt/cdrom　　在/mnt目录下建一个光驱的挂载点cdrom。如果你已经有cdrom这个挂载点，则可以略去这步。

mount /dev/cdrom /mnt/cdrom　　将光驱挂载到相应的挂载点cdrom上。

cd /mnt/cdrom/Tools/uclinux_tools　　进入相应的目录

./ arm-elf-tools-20030314.sh　　运行安装程序

这时我们的交叉编译环境就成功建立起来了。

第十一章 uClinux 的编译步骤

1、下载 uclinux

51EDA 的网站上为大家提供了已经根据豪华板的硬件进行设置的 uClinux，用户可以省去大量移植代码的工作。在配套光盘中，uClinux 的源代码存放在 Source 文件夹下

2、解压缩

tar -jxf uClinux20050122.tar.bz2

（注：随着软件的更新，51EDA 提供的 uClinux，其最后的版本号会有变化，解压时请注意分辨清楚）

解压完成后，用 ls 指令可以看到多出一个 uClinux-dist 的目录，如下图所示

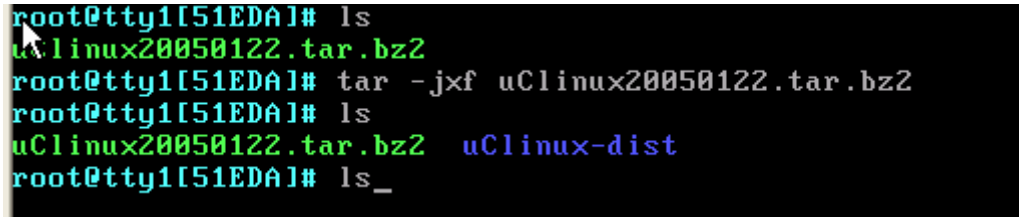


图 11-1

cd uClinux-dist　　进入 uclinux 的目录，下面的编译步骤均需要在这个目录下进行。

3、编译 uClinux 内核

如果用户需要订制自己的系统，那我们可以对uClinux 进行配置。通过 make menuconfig来实现。

对于编译uClinux，不能简单地通过make 来实现。我们需要有一些特定的步骤才能保证编译的正确。这是因为uClinux 所需要支持的硬件平台太多了，不能考虑的很周到。为了编译最后得到的镜像文件，我们需要Linux 的内核以及 romfs。对于我们的豪华板的移植来说，romfs 是被编译到内核里面去的。因此，

在编译内核前需要一个romfs。为了得到romfs 的image，我们又需要编译用户的应用程序。而为了编译用户的应用程序，我们又需要编译C 运行库，这里我们用的C 运行库是uClibc。

根据上面的分析，我们编译uClinux 的步骤如下：

```
make menuconfig
```

执行这条指令后，我们会遇到一个内核配置对话框，如下图如示：

图 11-2

我们先设置处理器的类型，在第一个选项处按回车，得到如下图所示的对话框

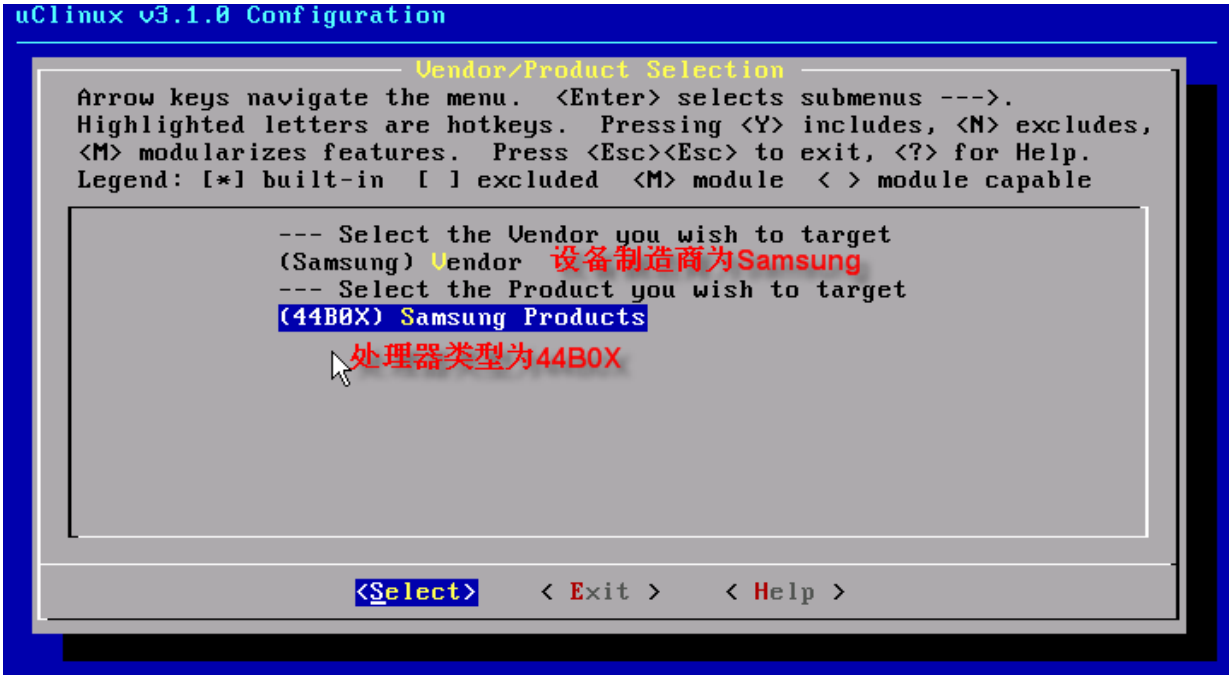


图 11-3

选择我们豪华板上的处理器类型，如上图所示：Samsung 44B0X，设置完成后，选择对话框底部的“Exit”退出至下一个界面。

图 11-4

这时，我们需要设置uclinux的内核，我们选择上图中的第二项，并回车。

这时会出现如下图所示的对话框

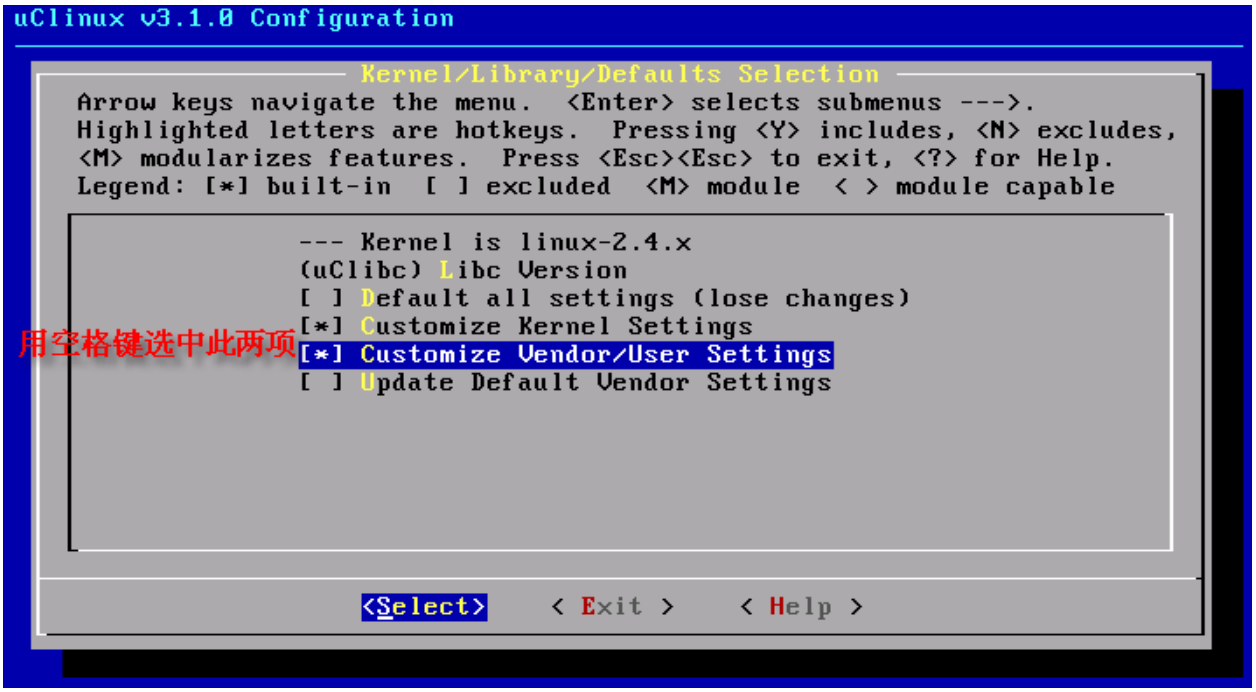


图 11-5

按上图所示，将中间的两项选中，使其[]中出现*。然后选中对话框底部的“Exit”，并回车。并再次回到了最初的设置对话框

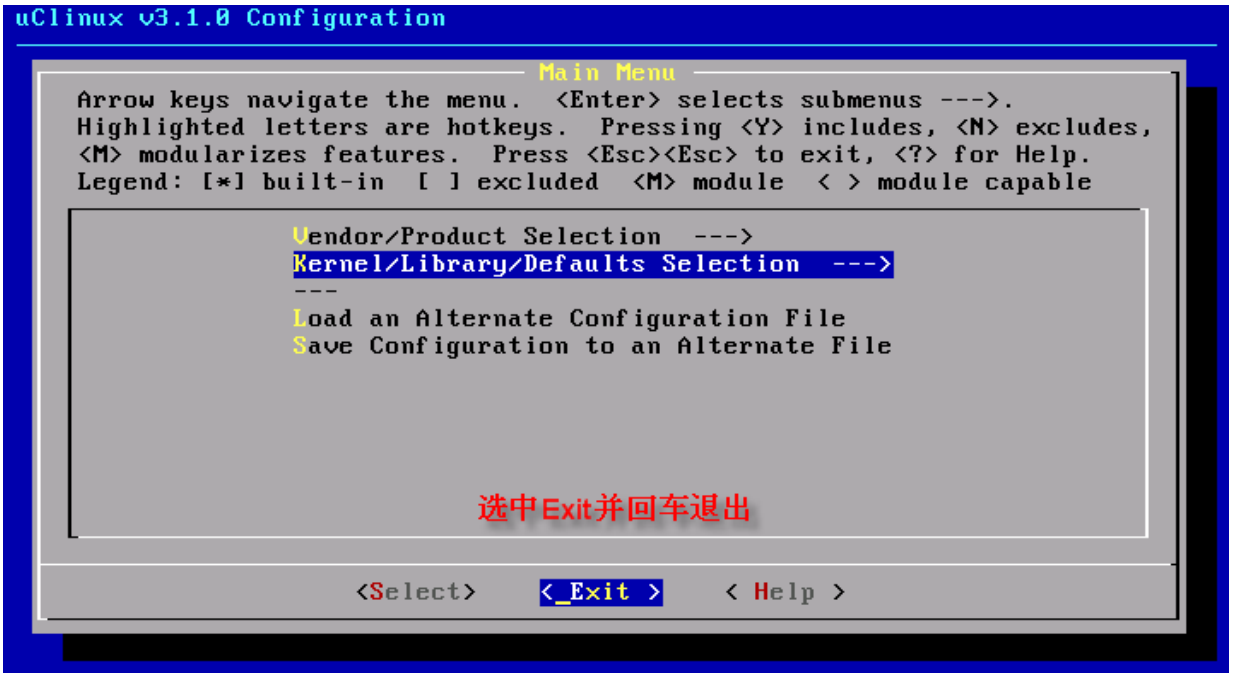


图 11-6

选中“Exit”回车后，退出。这时会出现一个对话框询问是否保存当前的内核设置，选择Yes，并回车。稍等一会，我们就进入了内核设置的界面。如下图所示

用户可以根据自己的需要，对内核进行设置，通常情况下可以略过，直接选择“Exit”退出。这时会跳出确认对话框。

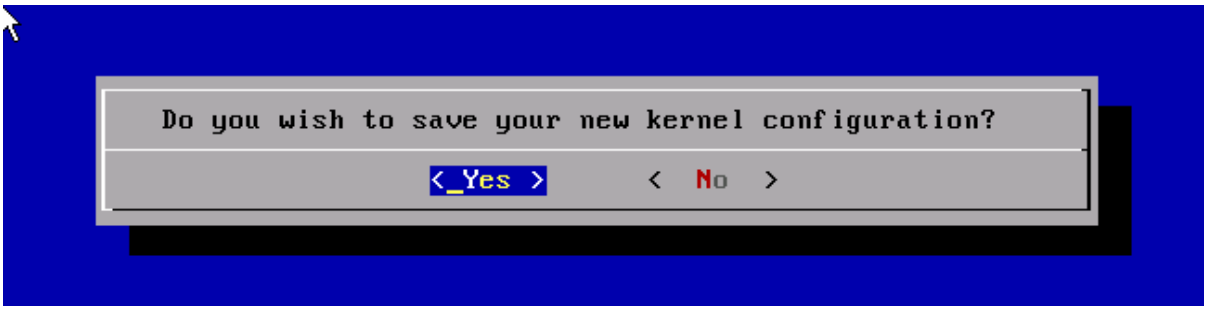


图 11-7

选择Yes并退出。稍等片刻，会进入应用程序的设置界面。

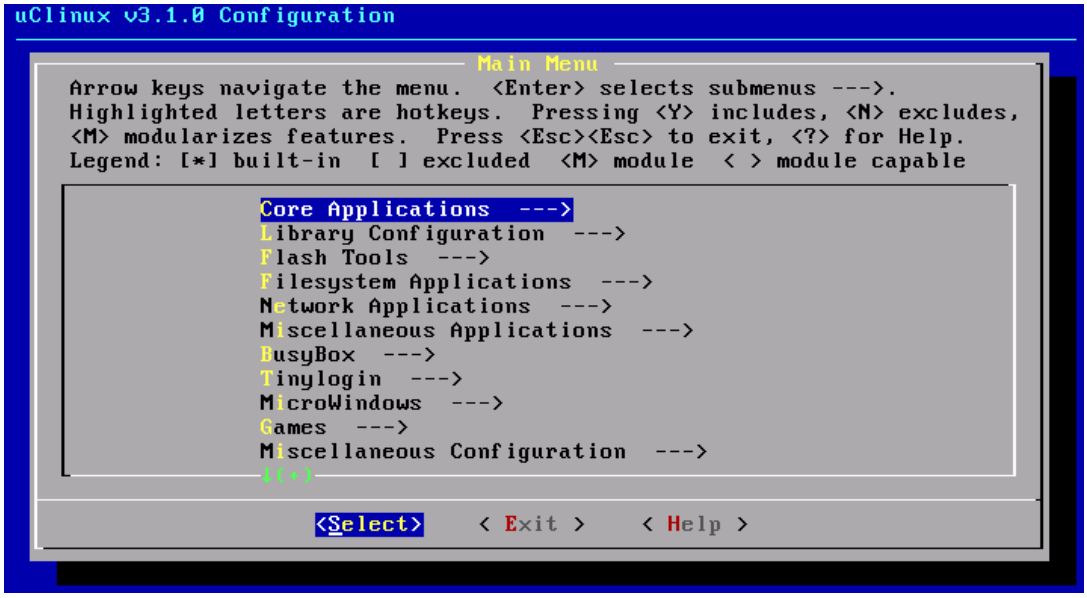


图 11-8

如果您对uclinux不太熟，建议您使用默认值，不需要更改，直接选择“Exit”并回车退出，在弹出的确认框中选择“Yes”。这时我们的内核设置工作就完成了。

make dep

这个仅仅是在第一次编译的时候需要，以后就不用了，为的是在编译的时候知道文件之间的依赖关系，在进行了多次得编译后，make 会根据这个依赖关系

来确定哪些文件需要重新编译、哪些文件可以跳过）。

make lib_only

编译uClibc。以后我们编译用户程序的时候需要这个运行库。

make user_only

编译用户的应用程序，包括初始化进程init，和用户交互的bash，以及集成了很多程序的busybox（这样对一个嵌入式系统来说可以减少存放的空间，因为不同的程序共用了一套C 运行库），还有一些服务，如boa（一个在嵌入式领域用的很多的Web 服务器）和telnetd（telnet 服务器，我们可以通过网络来登录我们的uClinux 而不一定使用串口）。

make romfs

在用户程序编译结束后，因为我们用到的是romfs（一种轻量级的、只读的文件系统）作为uClinux 的根文件系统，所以首先需要把上一步编译的很多应用程序以uClinux 所需要的目录格式存放起来。原来的程序是分散在user 目录下，现在例如可执行文件需要放到bin目录、配置文件放在etc 目录下……这些事就是make romfs 所做的。它会在uClinux 的目录下生成一个romfs 目录并且把user 目录下的文件、以及vendors 目录下特定系统所需要的文件（我们的vendors 目录是vendors/Samsung/44B0X）组织起来，以便下面生成romfs 的单个镜像所用。

make image

它的作用有2 个，一个是生成romfs 的镜像文件，另一个是生成Linux 的镜像。因为原来的Linux 编译出来是elf 格式的，不能直接用于下载或者编译（不过那个文件也是需要的，因为如果你需要，那个elf 格式的内核文件里面可以包含调试的信息）。因此在这个时候由于还没有编译过Linux，因此在执行这一步的时候会报错。但是没有关系，因为我们在这里需要的仅仅是romfs 的镜像，以便在下面编译Linxu 内核的时候使用。这步仅在第一次的时候出现，以后就不会出现了。

make linux

有了romfs 的镜像我们就可以编译Linux 了。因为我们的romfs 是嵌入到

linux 内核中去了，所以在编译Linux 内核的时候就要一个romfs.o 文件。这个文件是由上面的 make image生成的。

make image

这里再一次 make image 就是为了得到uClinux 的可执行文件的镜像了。执行了这一步之后，就会在images 目录下找到3 个文件：image.ram，image.rom，romfs.img。其中，image.ram和 image.rom 就是我们需要的镜像文件。

小结：在编译内核时，指令顺序分别为：

make menuconfig

make dep

make lib_only

make user_only

make romfs

make image (第一次执行该指令时会出错，可以不必理会)

make linux

make image

4、使用 uClinux

上面我们提到了两个文件：image.ram 和image.rom。其中，image.ram 是直接下载到RAM 执行的文件。如果你还处于调试阶段，那么就没有必要把文件烧写到FLASH 里面。这个时候我们可以使用image.ram。在开发板运行BIOS 的情况下，使用：

netrun

将文件下载到RAM 并且自动执行。具体可以参考BIOS 的教程。

对于image.rom 来说，它是一个zImage 文件，也就是自解压的内核。由于它使用了gzip将内核压缩过,所以可以减小文件的大小。这个image 应该烧写到FLASH 的0x10000 的位置，而不能直接下载到RAM 并执行。在开发板运行了uClinux 之后，我们可以通过串口看到很多输出，但是最后可以看到

Welcome to

```
____ _ _
/ __| ||_|
_ _| | | | _ ____ _ _ _ _
| | | | | | || | _ \| | | |\ \/ /
| |_| | |__| || | | | | |_| |/ \
| ___\____|_||_|_| |_|\____|\_/\_/
| |
|_|
```

For further information check:

http://www.uclinux.org/

This port can be run under 51EDA's ARM Development Board

http://www.51eda.com/

Execution Finished, Exiting

Sash command shell (version 1.1.1)

/>

这样就表明你的uClinux 正常启动了。这时候可以输入一些命令，例如：

```
/> ls -l
drwxr-xr-x 1 0 0 32 Jan 1 00:00 bin
drwxr-xr-x 1 0 0 32 Jan 1 00:00 dev
drwxr-xr-x 1 0 0 32 Jan 1 00:00 etc
drwxr-xr-x 1 0 0 32 Jan 1 00:00 home
drwxr-xr-x 1 0 0 32 Jan 1 00:00 lib
drwxr-xr-x 1 0 0 32 Jan 1 00:00 mnt
dr-xr-xr-x 16 0 0 0 Jan 1 00:00 proc
lrwxrwxrwx 1 0 0 97 Jan 1 00:00 sbin -> /bin
lrwxrwxrwx 1 0 0 105 Jan 1 00:00 tmp -> /var/tmp
drwxr-xr-x 1 0 0 32 Jan 1 00:00 usr
drwxr-xr-x 8 0 0 1024 Jan 1 00:00 var
```

/>

ls 命令可以列出当前目录下的文件，-l 参数可以列出更详细的信息。

/> cat /proc/meminfo

total: used: free: shared: buffers: cached:

Mem: 14630912 2199552 12431360 0 499712 507904

Swap: 0 0 0

MemTotal: 14288 kB

MemFree: 12140 kB

MemShared: 0 kB

Buffers: 488 kB

Cached: 496 kB

SwapCached: 0 kB

Active: 744 kB

Inactive: 240 kB

HighTotal: 0 kB

HighFree: 0 kB

LowTotal: 14288 kB

LowFree: 12140 kB

SwapTotal: 0 kB

SwapFree: 0 kB

/>

cat 可以将一个文件显示出来，这里的 /proc/meminfo 表示内存的信息。您还可以看看proc 目录下有些什么文件并且把他们显示出来看看

/> ps

PID PORT STAT SIZE SHARED %CPU COMMAND

1 S 134K OK 0.1 init

2 S OK OK 0.0 keventd

3 S OK OK 0.0 ksoftirqd_CPU0

```
4 S 0K 0K 0.0 kswapd
5 S 0K 0K 0.0 bdflush
6 S 0K 0K 0.0 kupdated
15 S0 R 136K 0K 0.0 /bin/sh
16 S 70K 0K 0.0 /bin/inetd
17 S 264K 0K 0.0 /bin/boa
/>
```

ps 命令列出当前的进程的情况

在上面ps 的输出结果我们可以但到里面有boa，这就是我们的Web 服务器。另外，telnetd

是通过inetd 来启动的。

先来看一下boa。我们在主机的浏览器输入 http://192.168.1.100

就应该看见一个默认的页面：

图 11-9

然后我们也可以通过telnet 来访问我们的uClinux 机器：

在主机使用Telnet 工具登录到uClinux（IP 地址上面有了，是192.168.1.100）

它会要你登录，选择用户名root，密码为空，就可以进去了。

就如下图所示：

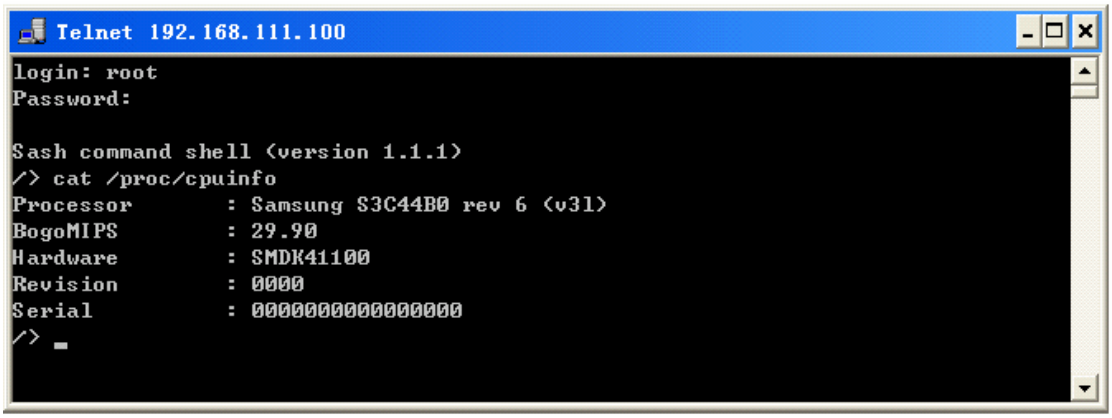


图 11-10

如果这时候用ps 查看的话，就会发现多出来了2 个进程：

PID PORT STAT SIZE SHARED %CPU COMMAND

1 S 134K OK 0.0 init

2 S OK OK 0.0 keventd

3 S OK OK 0.0 ksoftirqd_CPU0

4 S OK OK 0.0 kswapd

5 S OK OK 0.0 bdflush

6 S OK OK 0.0 kupdated

15 S0 R 140K OK 0.0 /bin/sh

16 S 70K OK 0.0 /bin/inetd

17 S 268K OK 0.0 /bin/boa

19 S 74K OK 0.0 /bin/telnetd

20 p0 S 134K OK 0.0 sh

下面的第19、20 号进程就是新的。也就是和telnet 有关的两个进程。

第十二章　uClinux 下驱动测试

1、按需要重编译 uClinux

为了正常使用开发板上的硬件资源，我们在编译 uClinux 的时候，要注意把需要使用到的驱动程序模块选上，并编译到内核中去，配套光盘里提供的 uClinux 源代码已经做好了各种驱动的配置，用户可以不需要重新进行配置（IDE 部分除外，因为大多数的用户平时并不使用 IDE 接口的硬盘，为了加快 uClinux 的启动速度，我们没有把 IDE 的驱动选中，如果需要使用的话，请根据第六小节的说明进行配置）。

2、网卡 CS8900A 测试

- 配置系统内核

正常情况下，这部分的配置在出厂之前就已经正确配置好，如果用户希望自己配置的话，请按上章的操作步骤，进入 uClinux 内核配置主界面，并选中下图中红框处的选项

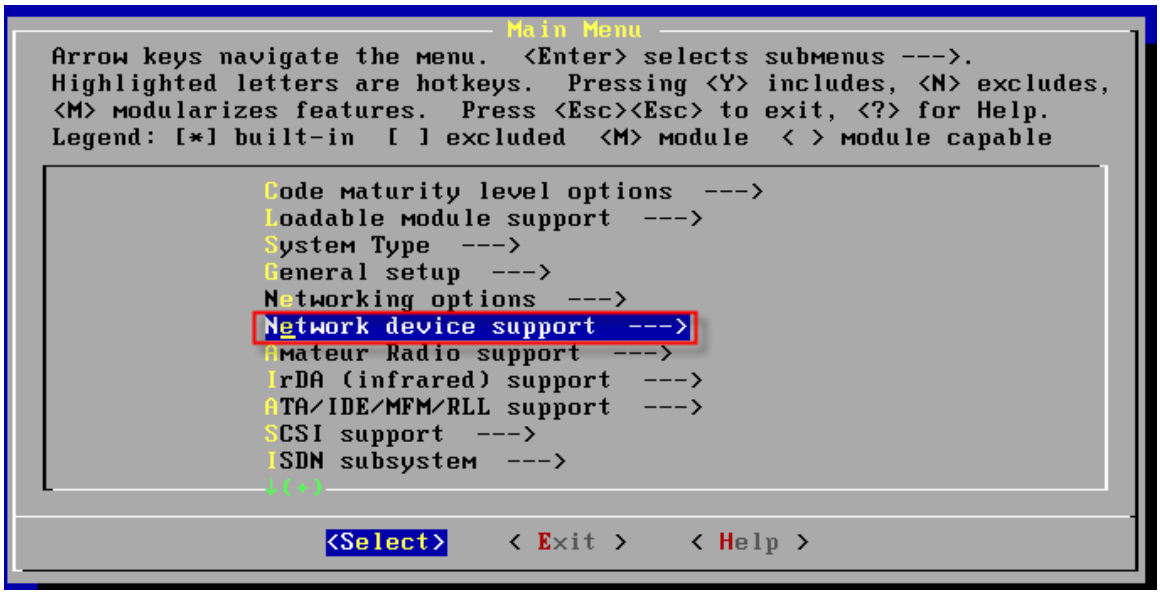


图　12-1

然后再按下图的提示，选中红框处的选项

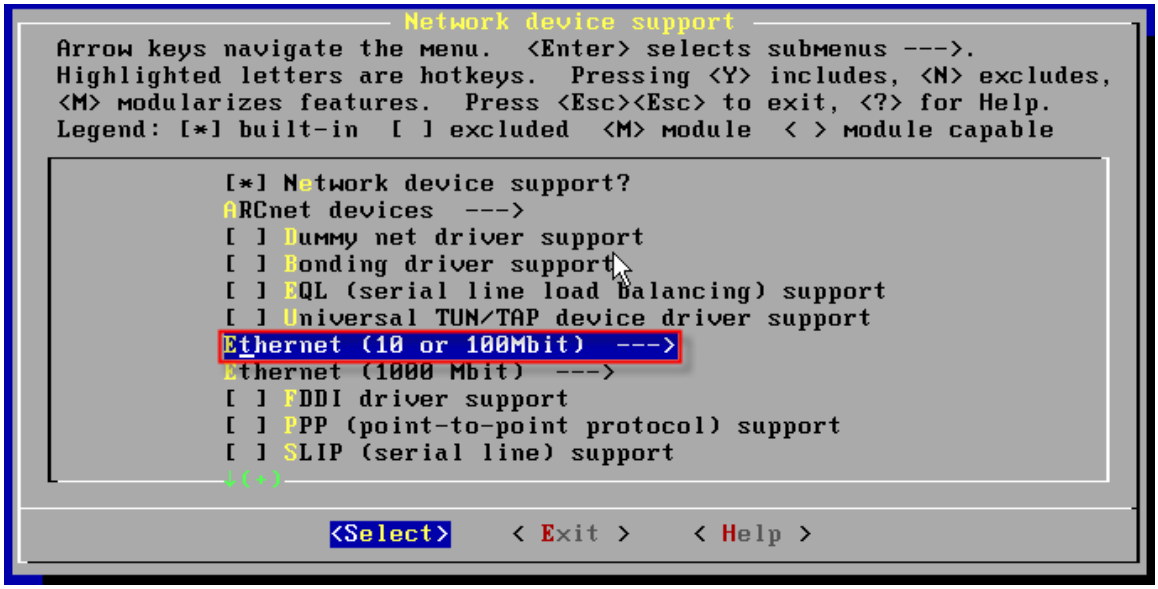


图 12-2

然后就可以看到 cs89x0 网卡驱动的选项

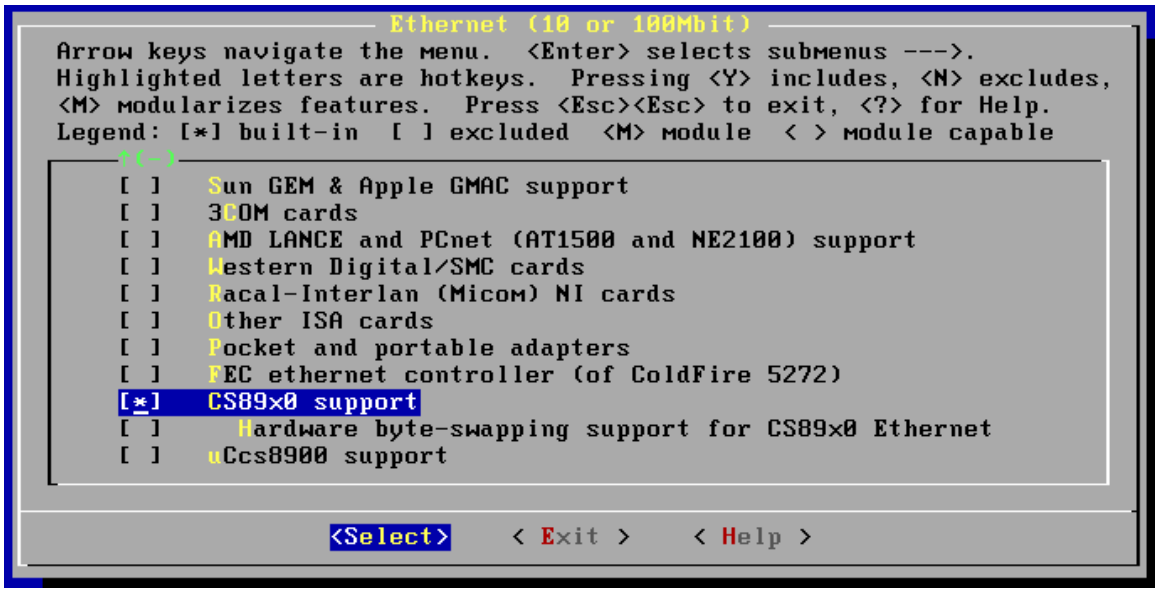


图 12-3

- 编译

执行下面的指令

./build_first.sh

在第一次执行./build_first.sh 时，会出错但这是正常的，无需理会出错信息

然后再执行

./build_final.sh

执行完后，uClinux 的编译就完成了，我们可以在 image/目录下找到生成的 image.ram 和 image.rom 文件。

- 测试

用交叉网线将开发板和 PC 机连接起来

将 image.rom 烧写到 FLASH 中，或把 image.ram 下载到 SDRAM 中后（具体步骤请参考上述章节），启动 uClinux，然后执行下面的指令

ping 192.168.1.x //这里的 IP 地址是你 PC 机的 IP 地址

当超级终端显示的提示信息如下图时，说明网卡工作正常：

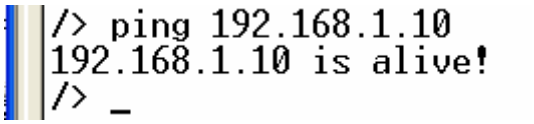


图 12-4

在 PC 机上执行下面的命令

ping 192.168.1.100 //这里的 IP 地址是开发板的 IP 地址

当出现如下提示信息时，说明网卡工作正常

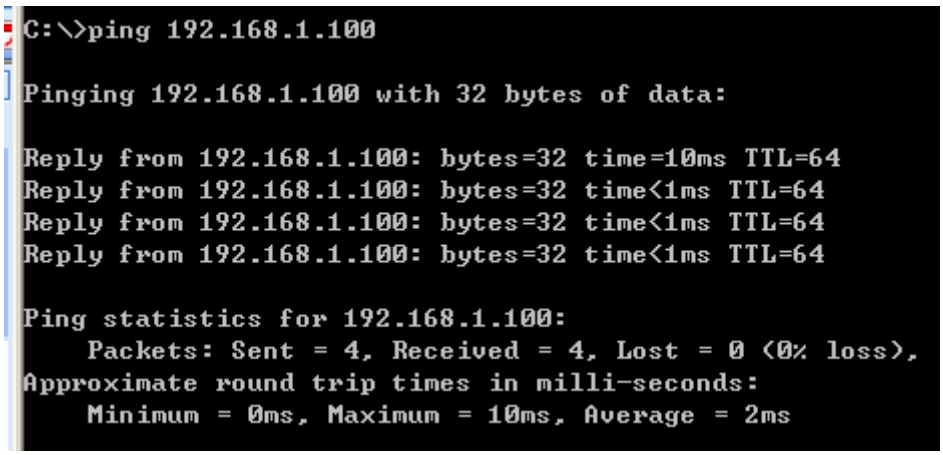


图 12-5

3、USB HOST

- 配置系统内核

正常情况下，这部分的配置在出厂之前就已经正确配置好，如果用户希望自己配

置的话，请按上章的操作步骤，进入 uClinux 内核配置主界面，并选中下图中红框处的选项

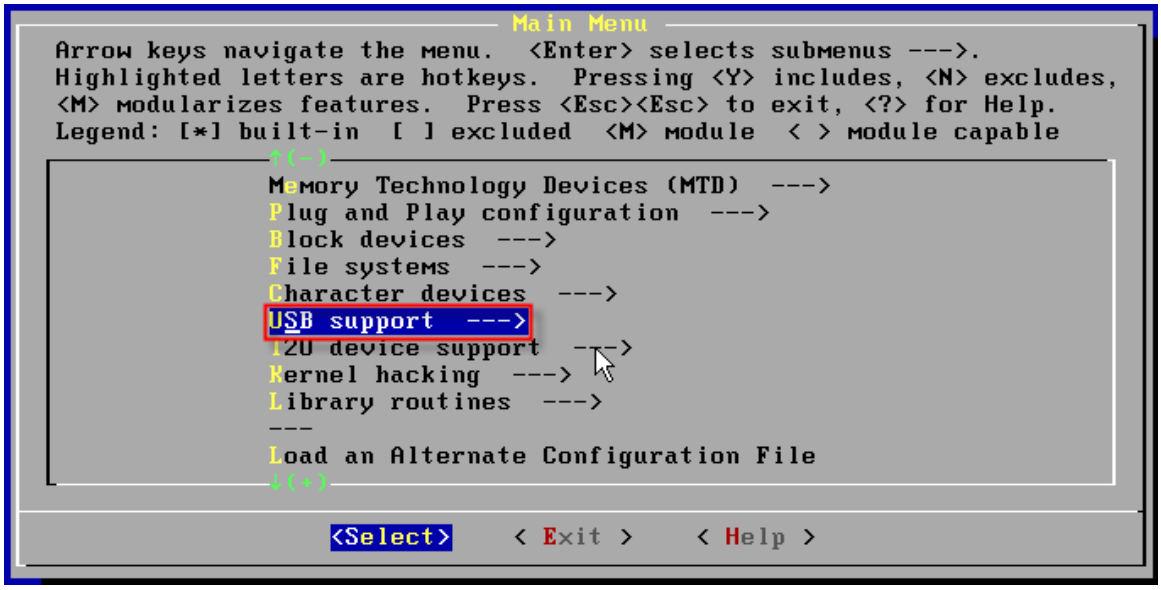


图 12-6

然后再按下图的提示，选中红框处的选项

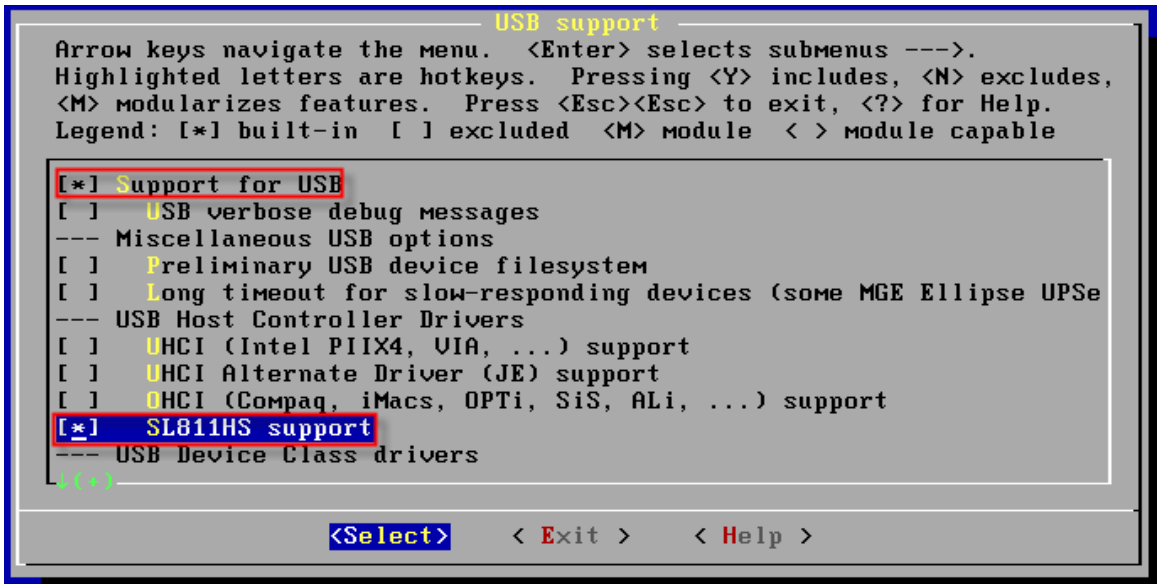


图 12-7

然后保存退出后，就完成了配置工作。

- 编译

  按上一小节的步骤进行编译

- 测试

image.rom 烧写到 FLASH 中，或把 image.ram 下载到 SDRAM 中后（具体步骤请参考上述章节），启动 uClinux。然后在插入 USB 设备，下图显示的是在开发板上插入罗技 USB 鼠标后的提示信息，因为开发板的 uClinux 没有该鼠标的驱动程序，因此 uClinux 只输出了该 USB 设备的 VendorID 和 ProductID，并提示没有驱动程序。

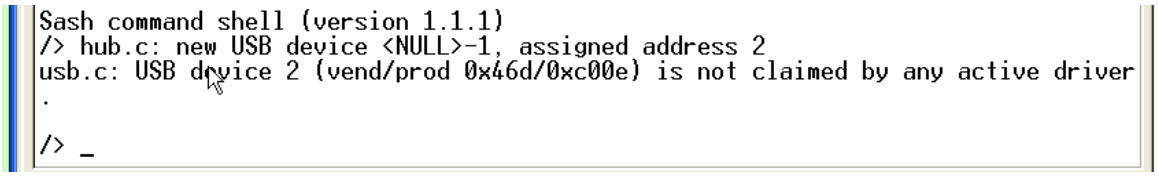


图 12-8

为了证明 USB host 已经驱动起来，我们可以用下面的指令来查看中断的响应情况。

cat /proc/interrupts

在下图中，我们可以看出 SL811 响应了 48 次中断请求，其中断号为 24。

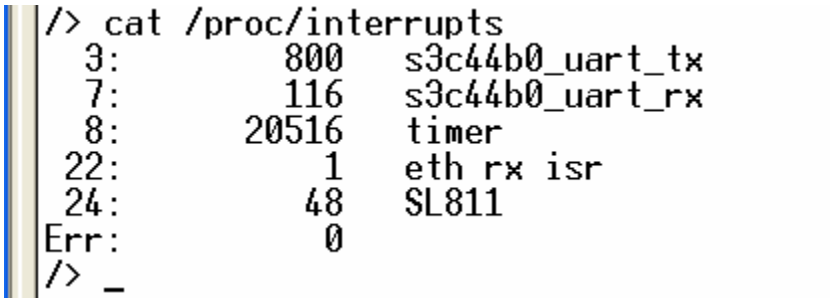


图 12-9

4、JFFS2 文件系统测试

- 配置系统内核

开发板上的 JFFS2 文件系统提供了一个 1M 大小的存储空间，存储在这个存储空间的内容掉电后可以保存，可以用于保存用户的文件。

正常情况下，这部分的配置在出厂之前就已经正确配置好，如果用户希望自己配置的话，请按上章的操作步骤，进入 uClinux 内核配置主界面，并选中下图中红框处的选项

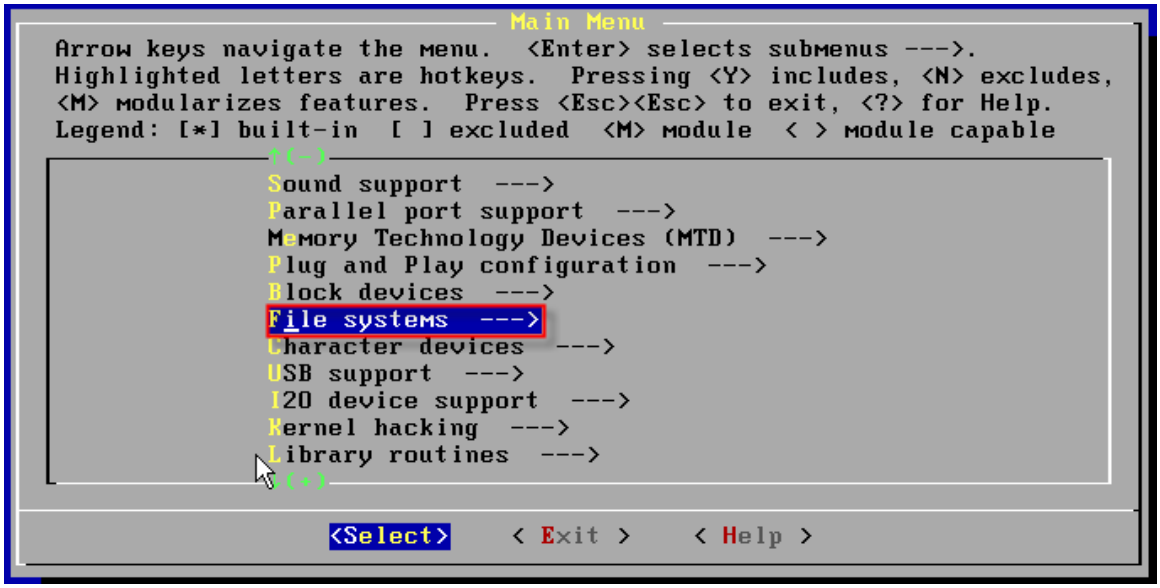


图 12-10

然后再按下图的提示，选中红框处的选项

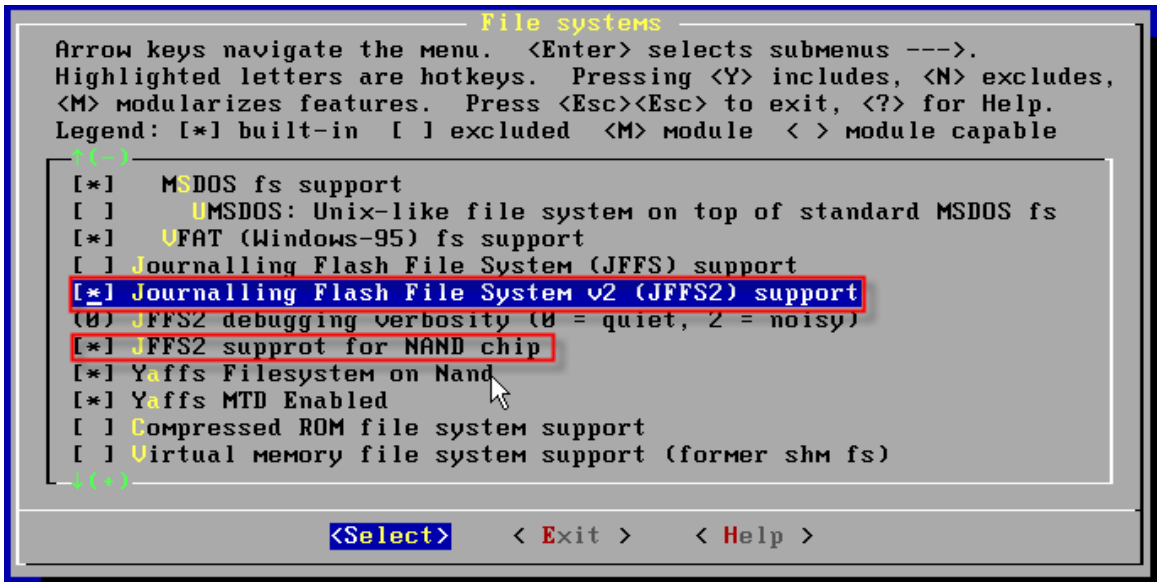


图 12-11

然后保存退出后，就完成了配置工作。

- 编译

  按上一小节的步骤进行编译

- 测试

image.rom 烧写到 FLASH 中，或把 image.ram 下载到 SDRAM 中后（具体步骤请参考上述章节），重启 uClinux 后，系统会自动把 jffs2 文件系统挂载到/mnt/jffs2

目录下。执行下面指令进入 jffs2 的挂载点：

```
cd /mnt/jffs2
```

这时就可以对 JFFS2 的存储空间进行操作了。

```
mkdir    test          //建立一个 test 的目录
ls                     //查看目录
```

测试画面如下，且开发板重启后，test 目录不会丢失

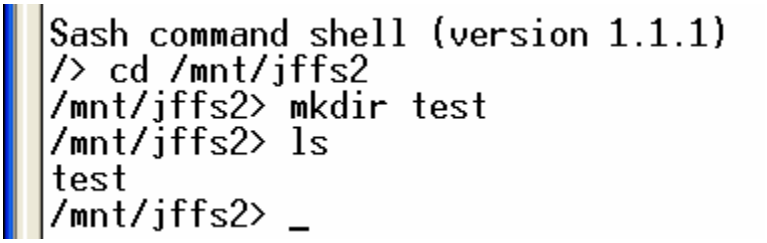


图 12-12

5、YAFFS 文件系统测试

- 配置系统内核

开发板上的 YAFFS 文件系统提供了一个 15M 大小的存储空间，存储在这个存储空间的内容掉电后可以保存，可以用于保存用户的文件。

正常情况下，这部分的配置在出厂之前就已经正确配置好，如果用户希望自己配置的话，请按上章的操作步骤，进入 uClinux 内核配置主界面，并选中下图中红框处的选项

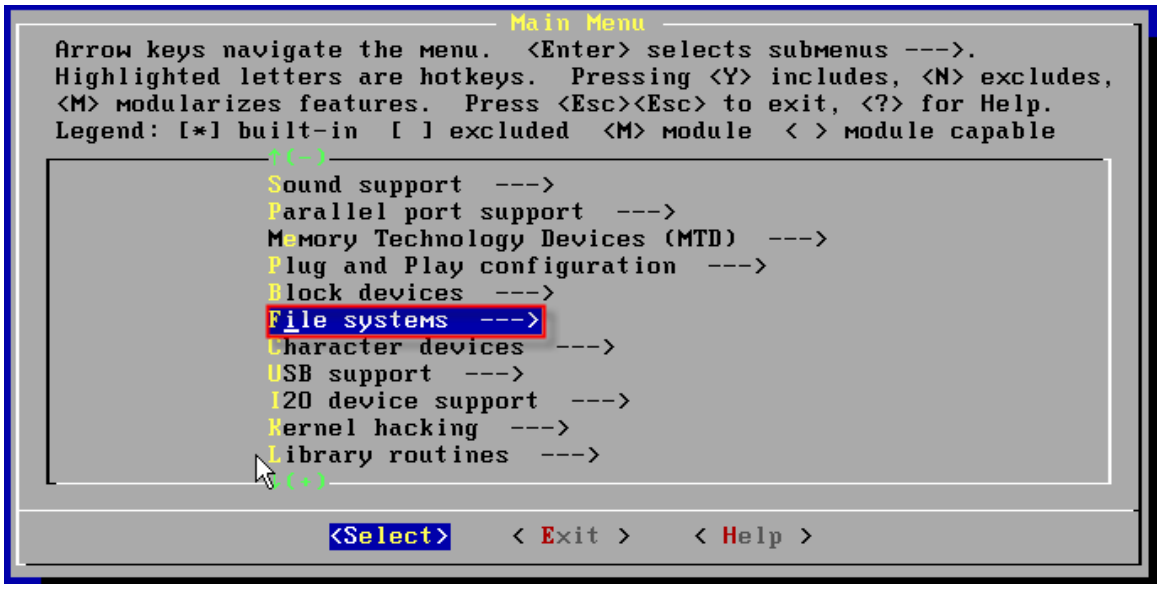


图 12-13

然后再按下图的提示，选中红框处的选项

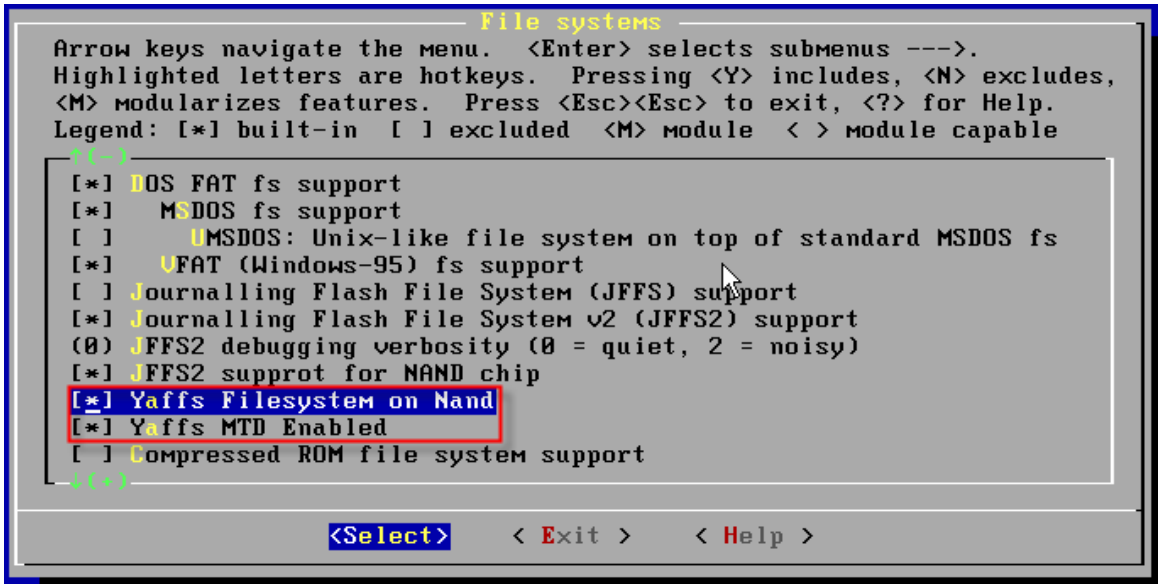


图 12-14

然后保存退出后，就完成了配置工作。

- 编译

按上一小节的步骤进行编译

- 测试

image.rom 烧写到 FLASH 中，或把 image.ram 下载到 SDRAM 中后（具体步骤请参考上述章节），重启 uClinux 后，系统会自动把 yaffs 文件系统挂载到/mnt/yaffs 目录下。执行下面指令进入 yaffs 的挂载点：

```
cd /mnt/yaffs
```

这时就可以对 JFFS2 的存储空间进行操作了。

```
mkdir    yaffs          //建立一个 yaffs 的目录
ls                      //查看目录
```

测试画面如下，且开发板重启后，yaffs 目录不会丢失

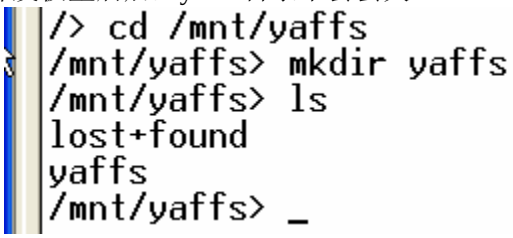


图 12-15

6、IDE 硬盘测试

- 系统连接图

按下图的连接位置连接硬盘，硬盘的电源须外接。

图 12-16

- 配置系统内核

按上章的操作步骤，进入 uClinux 的内核配置主界面

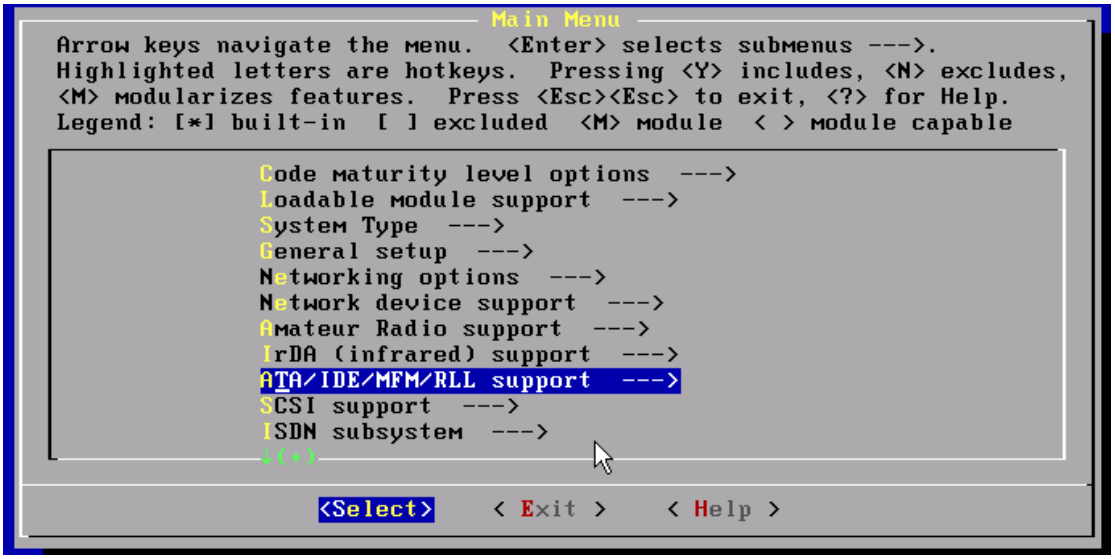


图 12-17

在上图的 MainMenu 中，我们选择光标所在位置的选项 ATA/IDE/MFM/RLL support，并回车

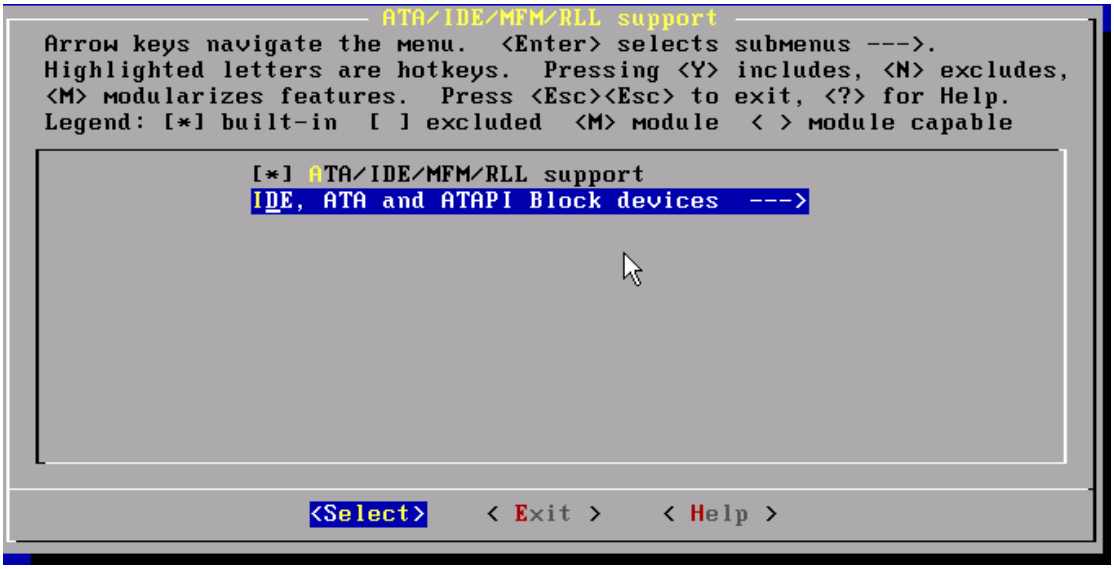


图 12-18

用空格选中上图中带*的选项，并选择光标所在的 IDE, ATA and ATAPI Block devices，回车

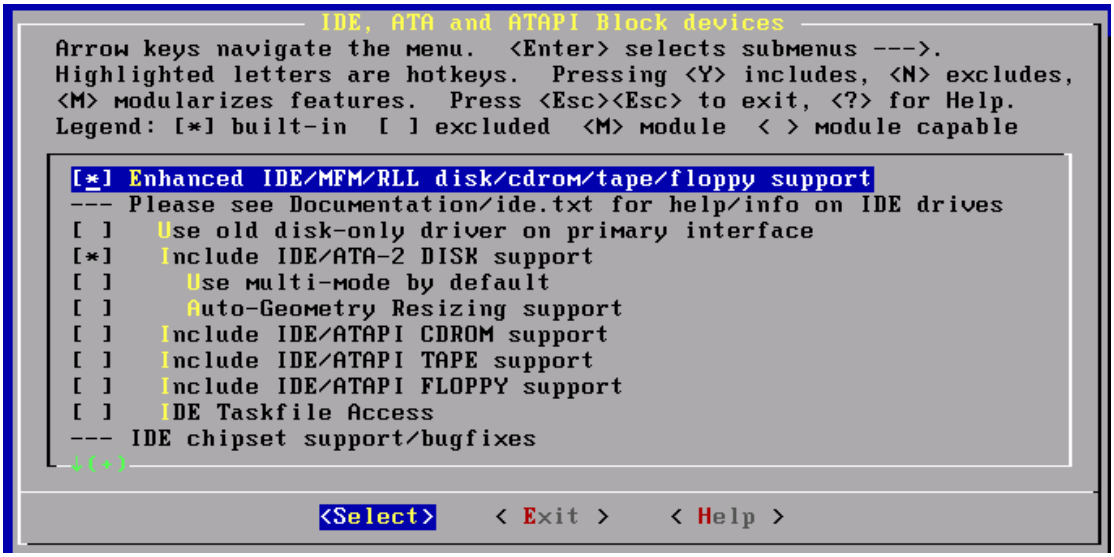


图 12-19

按上图所示，选中带*的选项。并选两次 Exit，回到 MainMenu

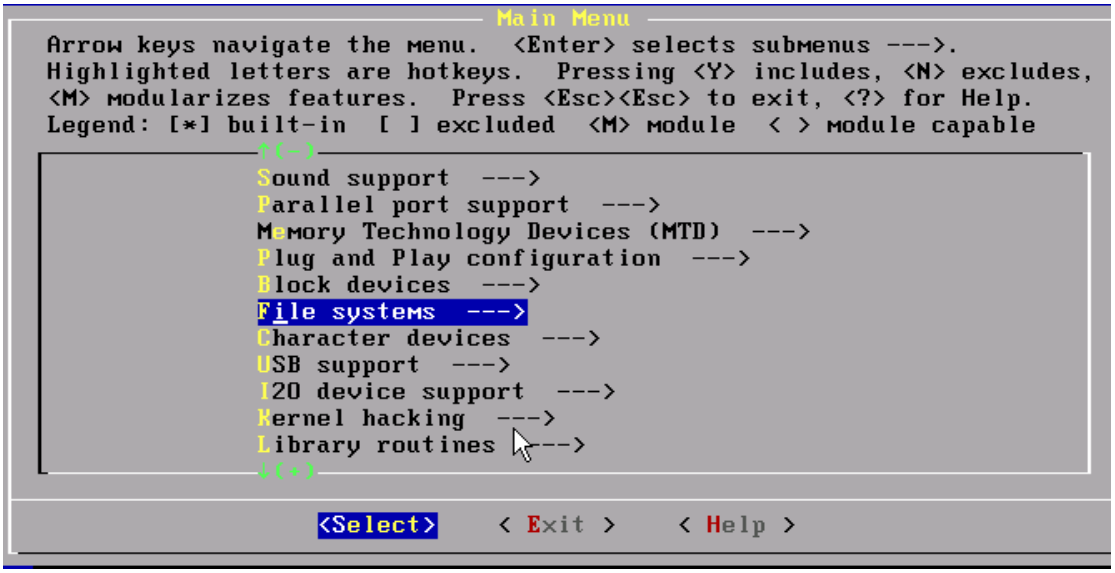


图 12-20

选择光标所在的选项 File systems，并回车

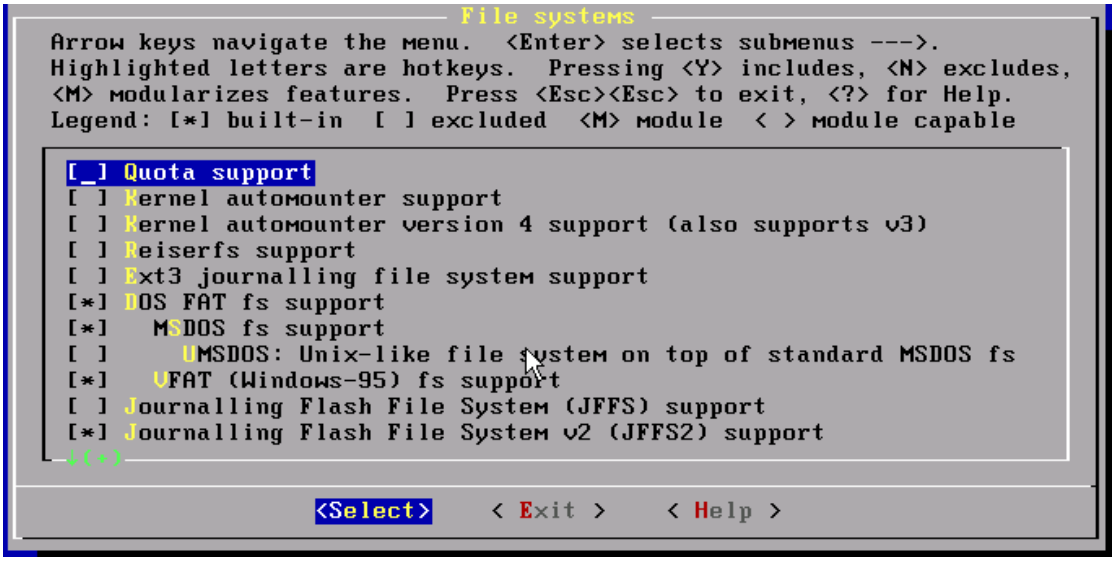


图 12-21

按上图用空格选中带*的选项，并往下移动光标，看到如下选项时，选择 Partition Types

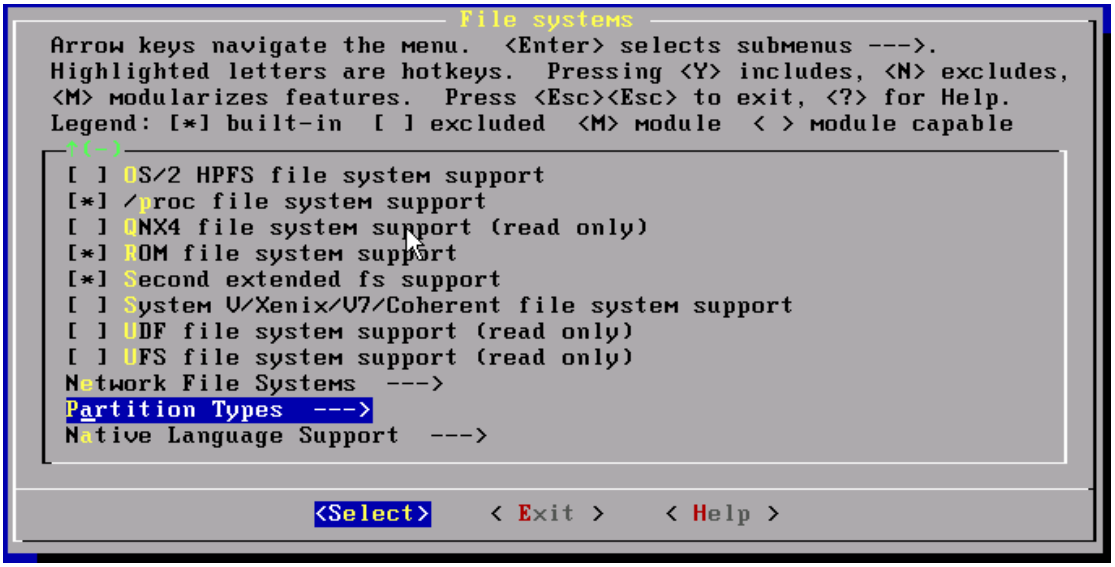


图 12-22

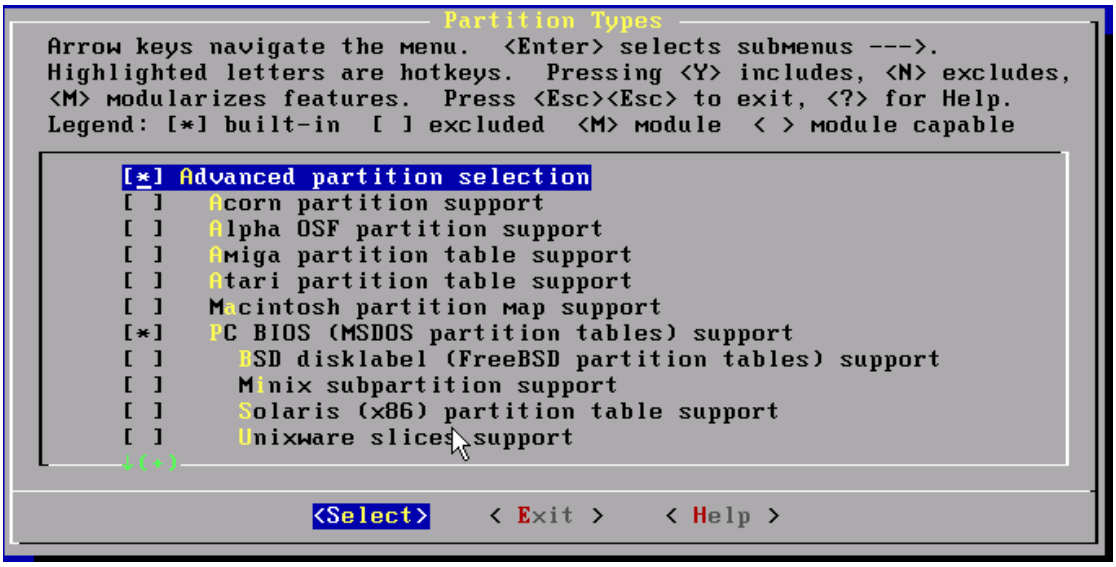


图 12-23

用空格选中带*的选项，三次选择 Exit 后

图 12-24

选择 Yes，至此，我们已经完成了内核的配置工作，下面开始编译

- 编译

执行下面的指令

./build_first.sh

在第一次执行./build_first.sh 时，会出错，但这是正常的，无需理会出错信息

然后再执行

./build_final.sh

执行完后，uClinux 的编译就完成了，我们可以在 image/目录下找到生成的 image.ram 和 image.rom 文件。

- 烧写 image.rom

注意：在使用 IDE 硬盘时，我们不能使用 ram 中运行的 image.ram。原因如下：在通过网络下载 uclinux 的映象文件时，需要断开硬盘，如果下载完成后，我们再把硬盘接上或给硬盘通电，这时硬盘没有收到一个 RESET 信号，不能正确初始化。如果我们直接运行，则不能正确识别硬盘。硬盘的这个 RESET 信号是由豪华板的 RESET 信号提供的，如果我们复位重启豪华板，则 RAM 的映像文件会丢失。所以我们只能使用 image.rom。烧写到板上后，只要在主板通电前把硬盘接上并通电，硬盘都可以正确识别。

把 image.rom 烧写到 FLASH 的步骤如下：

1、断开硬盘(或不给硬盘供电)

2、进入豪华板的 BIOS，运行下载烧写指令：

appprog

3、主机端将 image.rom 文件通过网络发给目标板

4、根据提示，按”Y”键后烧写至 flash 中。

5、接上硬盘，给硬盘通电。

6、按 RESET，重启豪华板

7、启动后，会看到你的硬盘信息，如下如示，其中红色信息说明正确识别了我的硬盘，昆腾火球 4.3G 硬盘。如果你启动时没有看到你的硬盘信息，请换一块容量

稍小的硬盘进行测试。本驱动程序正常情况下可以驱动 60G 以下的硬盘，只支持 FAT 和 FAT32 分区格式，且不能正确识别中文文件名，可正确识别文件中的中文。

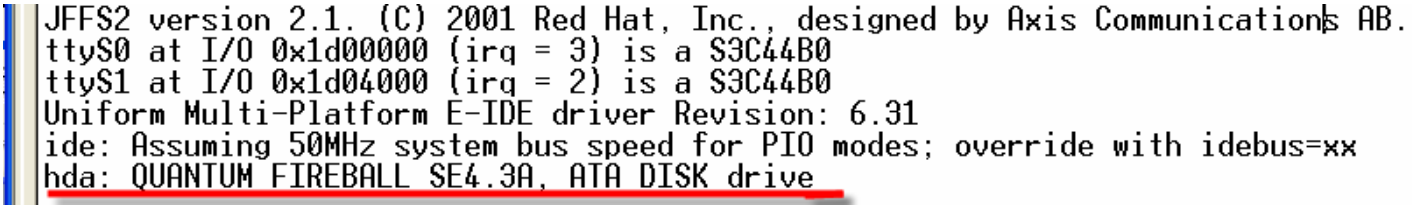


图 12-25

- 挂载硬盘

在使用 IDE 硬盘之前，我们需要把它挂载到我们的 uClinux 操作系统上，请执行如下指令：

mount -t vfat /dev/hda1 /mnt/hda1

这时，IDE 硬盘的第 1 个分区就挂载到开发板/mnt/hda1 的目录下了，如果需要挂载别的分区，请参阅相关的 linux 文档。

- 功能测试

1、列目录，执行

ls /mnt/hda1

可以看到硬盘中的目录，如下图所示

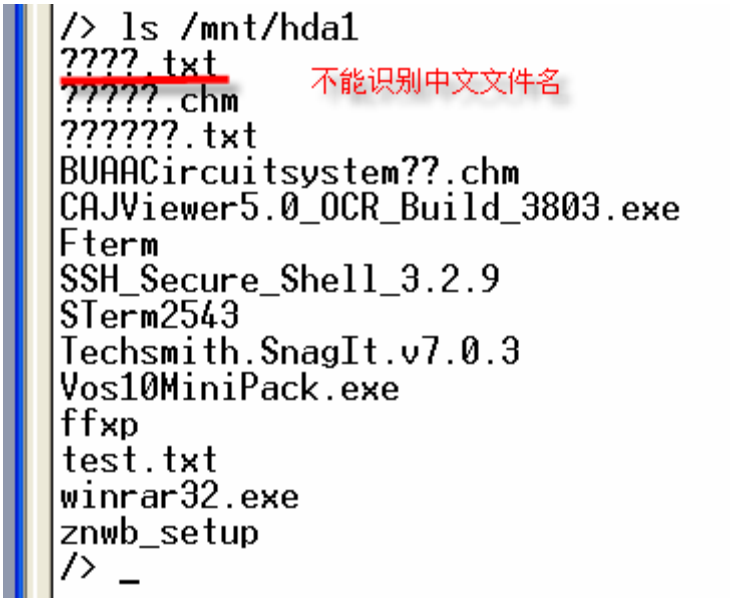


图 12-26

2、打开文件，执行

cat /mnt/hda1/test.txt

输出结果如下：(可以正确识别文件中的中文字符)

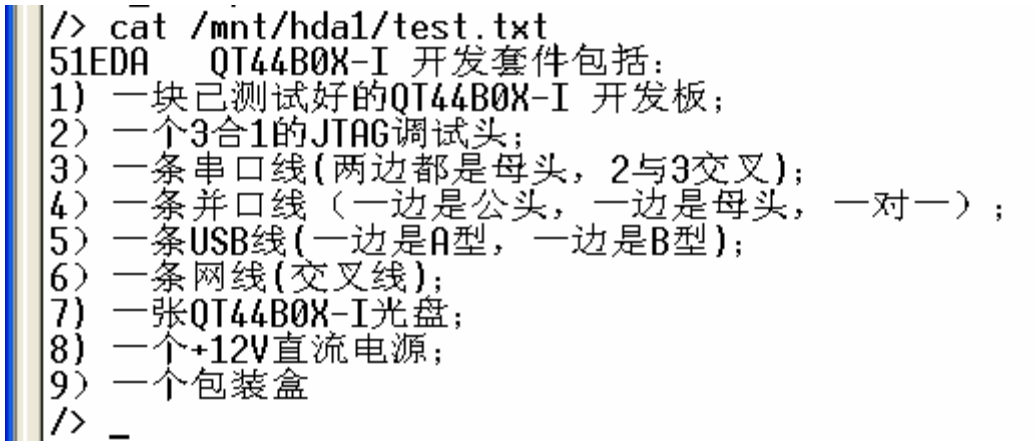


图 12-27

3、写测试，执行

mkdir /mnt/hda1/51eda

ls /mnt/hda1

结果如下：(51eda 目录已经被正确建立)

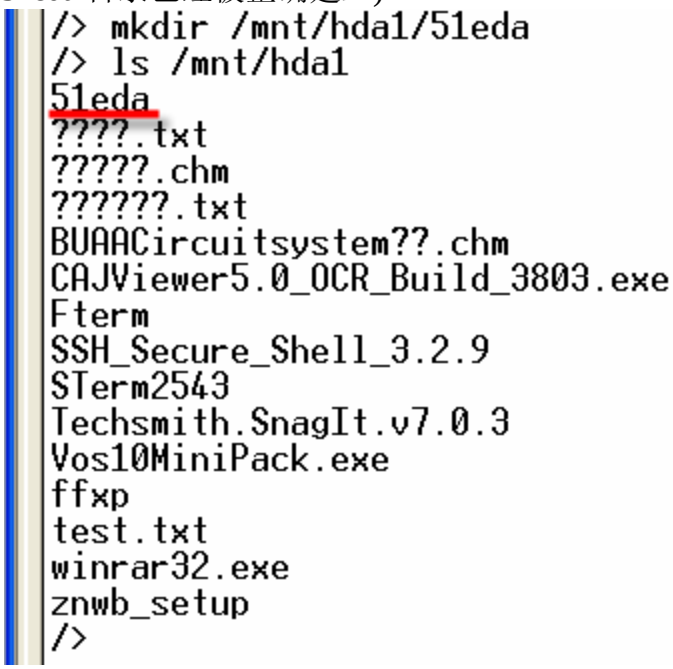


图 12-28

4、删除测试，执行

rmdir /mnt/hda1/51eda

ls /mnt/hda1

结果如下：(51eda 目录已经被删除)


/> rmdir /mnt/hda1/51eda
/> ls /mnt/hda1
????.txt
?????.chm
??????.txt
BUAACircuitsystem??.chm
CAJViewer5.0_OCR_Build_3803.exe
Fterm
SSH_Secure_Shell_3.2.9
STerm2543
Techsmith.SnagIt.v7.0.3
Vos10MiniPack.exe
ffxp
test.txt
winrar32.exe
znwb_setup
/> _

图 12-29

第十三章　NFS 文件系统的配置与使用

1、NFS 文件系统简析

在介绍 NFS 之前，我们先介绍挂载的概念。挂载就是把一个分区或目录映射到另一个目录(称为挂载点)上，之后对挂载点的操作将会映射到被挂载的分区或目录上。被挂载的分区或目录可以位于网络上的其它电脑中。比如说我们将 PC 机的/user 目录挂载到开发板的/mnt 目录下，那么开发板上的/mnt 目录实际上就是 PC 机上的/user 目录,你在开发板/mnt 目录下进行的操作都相当于对 PC 机/user 目录进行的。

NFS(network file system，网络文件系统)是 UNIX 类操作系统共享文件和应用程序的方法。NFS 和在 Windows 上通过网上邻居共享文件夹相似，它允许用户像访问自己的硬盘一样访问其它电脑上的硬盘或其它存储设备。如果我们让 PC 主机做 NFS 服务器，在目标系统上挂载主机提供的 NFS 文件夹，那么我们在宿主机上为目标系统编译的程序就不用每次都下载到目标系统去执行了，我们只需把要执行的文件复制到提供 NFS 服务的文件夹下，然后在目标系统上直接运行就可以了。使用 NFS 文件系统，为我们开发、调试嵌入式应用程序提供了极大的便利。

2、NFS 文件系统的配置

- 主机端配置

我们以比较通用的 RedHat 9.0 版本来举例说明 PC 主机端的 NFS 配置

在安装主机端的 Linux 时，要注意把所有的安装包都选上（如下图所示），以避免在后述步骤中遇到令人烦恼的库依赖问题

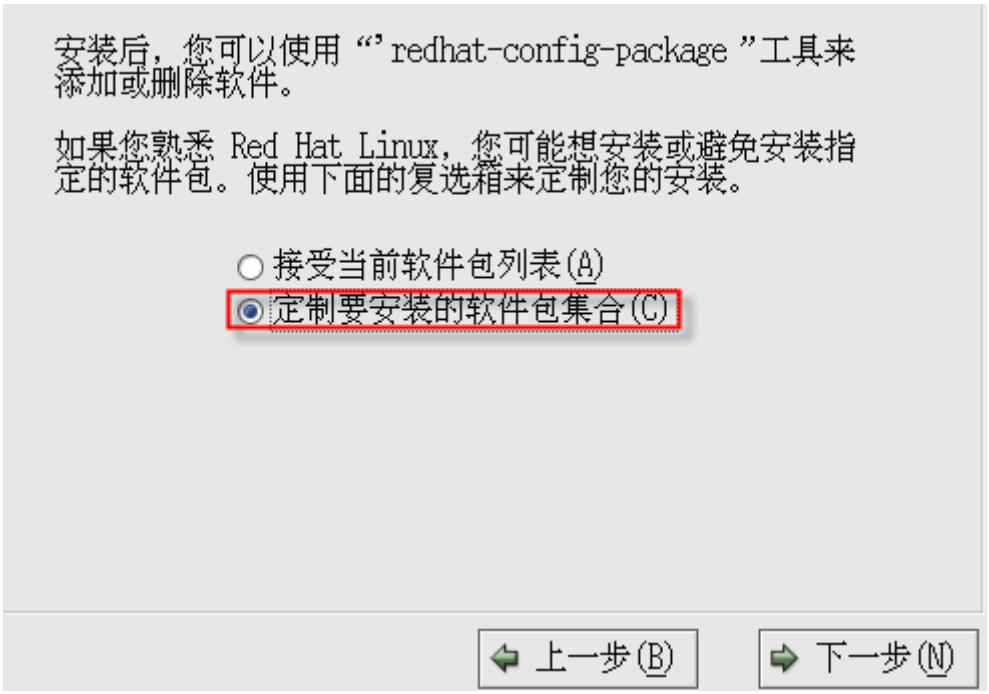


图 13-1

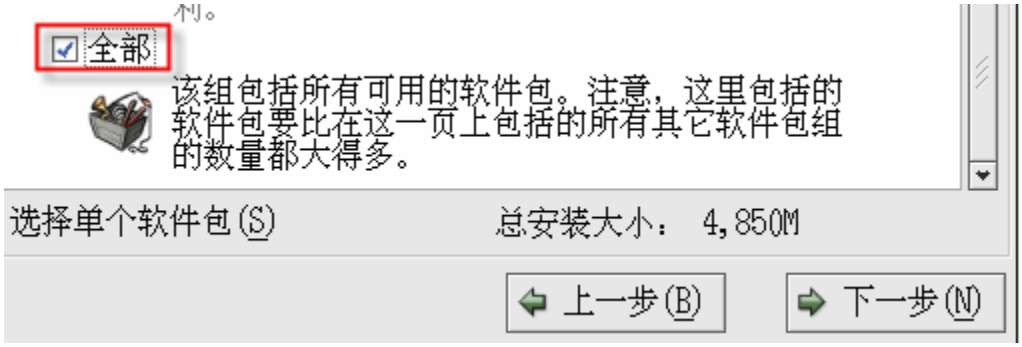


图 13-2

Linux安装完毕后，我们需要进行相关的设置，在命令终端窗口输入指令进行安装后的设置工作，以下操作均需要root权限

```
setup
```

此时，我们会看到如下图的设置界面

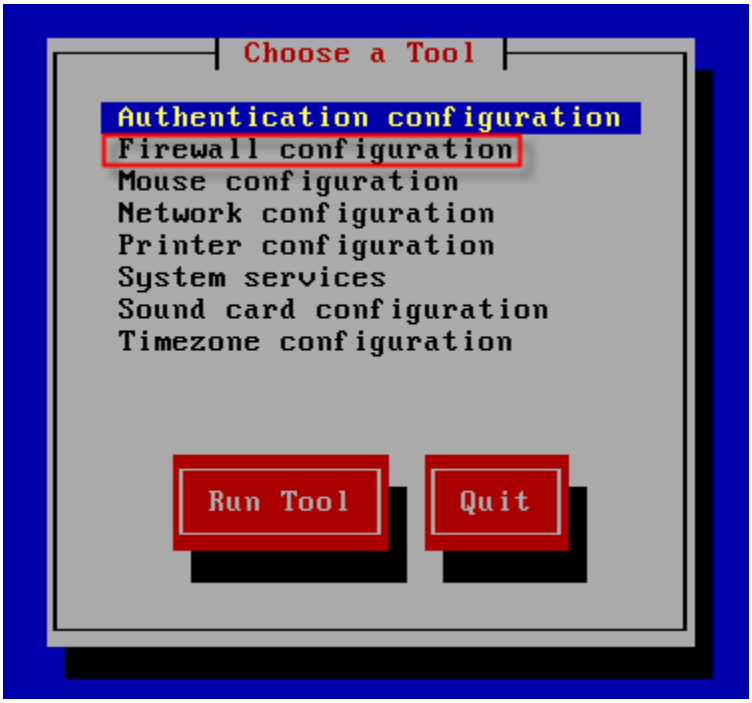


图 13-3

因为RedHat 9.0它默认的是打开了防火墙，因此对于外来的IP访问它全部拒绝，这样其它网络设备根本无法访问它，即开发板无法用NFS的方式mount上PC主机，因此系统安装完毕后，应关闭防火墙，选中上图红框处的选项，按下图的要求选择No Firewall：

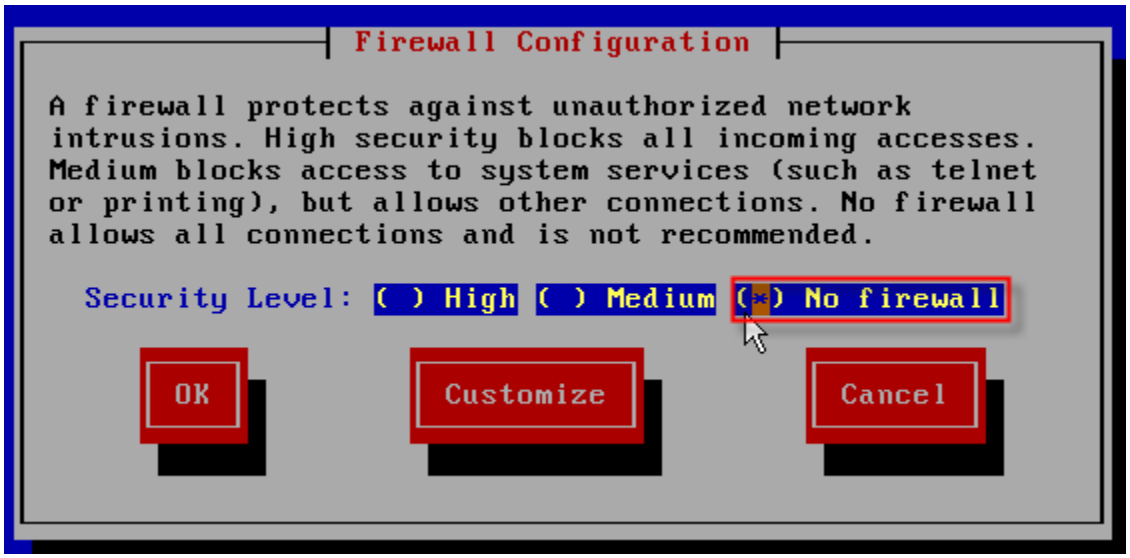


图 13-4

点击“OK”按钮后，回到主菜单，选择下图红框处的选项，配置系统服务：

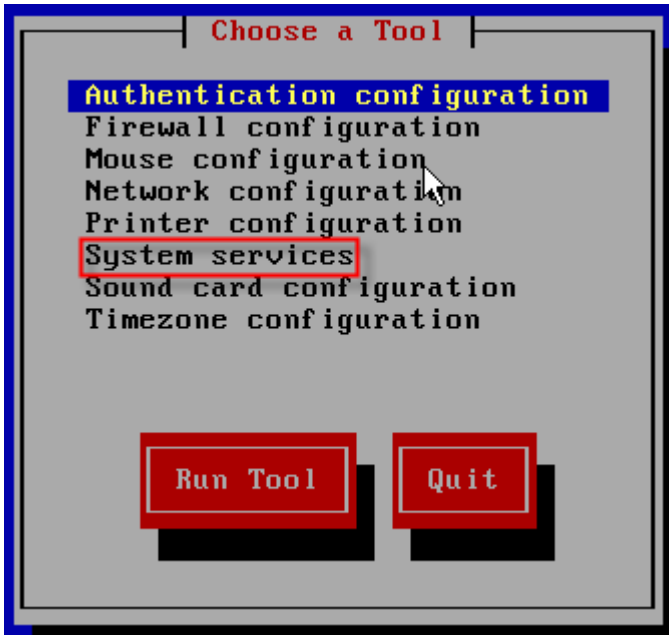


图 13-5

为了关掉防火墙，我们要把下图中的iptables和ipchains去掉。

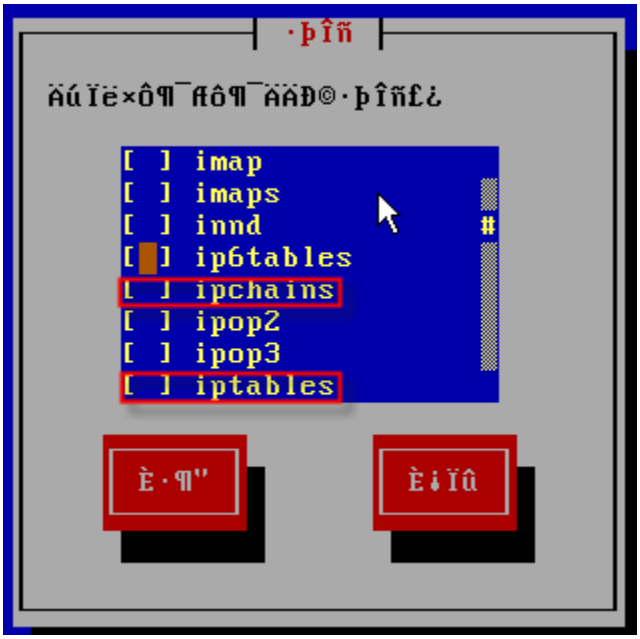


图 13-6

启动nfs系统服务，包括以下两个部分nfs和portmap：

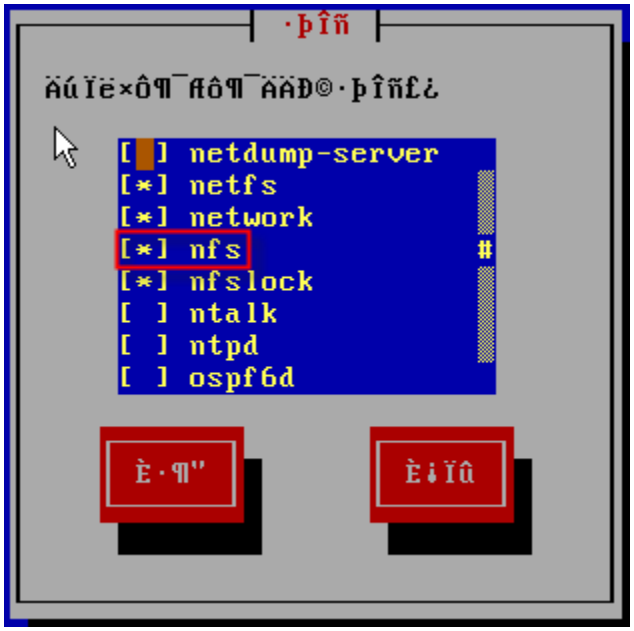


图 13-7

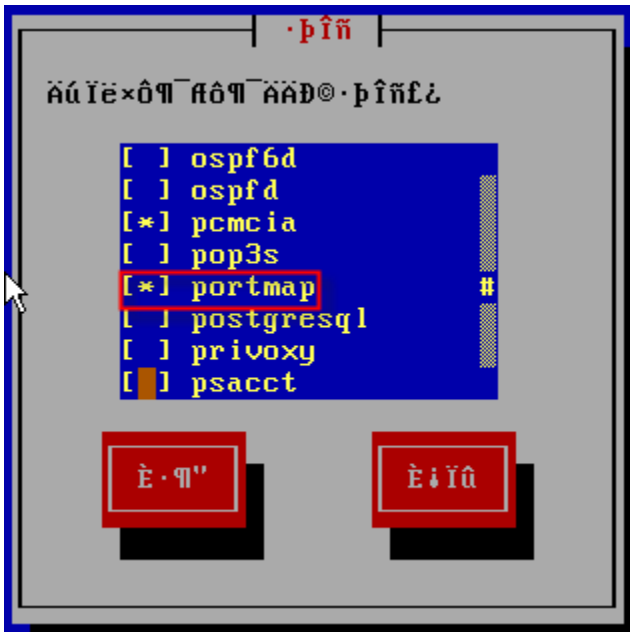


图 13-8

保存设置并退出到终端窗口，下面我们需要设置 PC 主机的共享目录，也就是告诉操作系统，主机上的那些目录是可以通过 NFS 进行远程访问的，通过修改/etc/exports 文件来实现，设定格式如下：

欲共享出去的目录 主机名称或者 IP(参数 1，参数 2)

上面这个格式表示，一个目录共享给指定的客户机，客户机的权限由参数所决

定。

可以设定的参数主要有以下这些：

rw：可读写的权限；

ro：只读的权限；

no_root_squash：连接到 NFS 主机的用户如果是 root 用户，他就拥有 root 的权限。

root_squash：当连接到 NFS 主机的使用者如果是 root，则该用户的 UID 和 GID 都为 0，即授予 nobody 的访问权限；

all_squash：不管连接 NFS 主机的用户是什么都会被重新设定为 nobody。

anonuid：将连接 NFS 主机的用户都设定成指定的 user id,此 ID 必须存在于/etc/passwd 中；

anongid：同 anonuid ，但是 user ID 为 group ID；

sync：资料同步写入存储器中。

async：资料会先暂时存放在内存中，不会直接写入硬盘。

insecure: 允许从这台机器过来的非授权访问。

通常我们常用的参数为 rw，ro 和 async

例如可以编辑/etc/exports 为：

/tmp　　*(rw，async)

表示主机上的/tmp 目录可以被任何一台客户机连接，权限为可读写

/uClinux-dist　　192.168.1.99(ro,async)

表示主机上的/uClinux-dist 目录能被 192.168.1.99 的客户机访问，权限只读

/uClinux-dist/user/hello　192.168.1.*(rw,no_root_squash,async)

表示主机上的/uClinux-dist/user/hello 目录能被 192.168.1.*客户机访问，权限是可读写

设定好后可以重启 Linux 或使用以下命令启动 NFS:

/etc/rc.d/init.d/portmap start

/etc/rc.d/init.d/nfs start

NFS 服务启动完成后，可用如下办法简单测试一下 NFS 是否配置好了：在 PC 主

机上自己 mount 自己，看是否成功就可以判断 NFS 是否配好了。执行：

mount 192.168.1.12:/tmp /mnt（此处 IP 为 PC 主机自己的 IP）

然后到/mnt 目录下看是否可以列出/tmp 目录下的所有文件和目录，可以则说明 mount 成功，NFS 配置成功。

这时我们就可以远程通过 NFS 来挂载主机上的目录了。

exportfs 命令：

如果我们在启动了 NFS 之后又修改了/etc/exports，是不是还要重新启动 nfs 呢？这个时候我们就可以用 exportfs 命令来使改动立刻生效，该命令格式如下： exportfs [-aruv]

-a ：全部 mount 或者 unmount /etc/exports 中的内容

-r ：重新 mount /etc/exports 中共享出来的目录

-u ：umount 目录

-v ：在 export 的时候，将详细的信息输出到屏幕上。

例：

[root @test root]# exportfs -rv <==全部重新 export 一次!

- **客户端（开发板）设置**

配套光盘中提供的 uClinux 已经为 mount 命令选上了 nfs 模块的支持，因此我们可以直接挂载 PC 主机上的目录

为了方便用户使用，我们在 uClinux 的 mnt 目录下建立了一个 nfs 的目录，用户可以把 PC 主机上的目录挂载到这个目录下。例如，我们希望把 PC 主机上/uClinux-dist/user 目录挂载到开发板/mnt/nfs 目录下，执行以下指令：

mount -t nfs 192.168.1.12:/uClinux-dist/user /mnt/nfs

这时我们就可以通过访问开发板上的/mnt/nfs 目录来远程访问 PC 主机上的/uClinux-dist/user 目录了。

- **功能测试**

在成功挂载 PC 主机的目录后，我们对/mnt/nfs 目录进行下列操作

cd /mnt/nfs　进入开发板的/mnt/nfs 目录

ls　查看目录文件

从下图中，我们就可以清楚地看到 PC 主机上的目录/uClinux-dist/user/hello 目录已经成功地挂载到开发板/mnt/nfs 目录下了。

```
Sash command shell (version 1.1.1)
/> mount -t nfs 192.168.1.12:/uClinux-dist/user/hello /mnt/nf
/> cd /mnt/nfs
/mnt/nfs> ls
Makefile
hello.c
/mnt/nfs> _
```

图 13-9

这时我们在 PC 主机编译 uClinux，编译完成后，我们再用 ls 命令，可以看到下图中的输出，这时可以看到 hello 程序已经被编译了，而我们也不需要把 image.ram 下载到板上才能调试 hello 程序了，我们可以直接在 nfs 文件夹下执行 hello 程序。

```
/> mount -t nfs 192.168.1.12:/uClinux-dist/user/hello /mnt/nfs
/> cd /mnt/nfs
/mnt/nfs> ls
Makefile
hello.c
/mnt/nfs> ls
Makefile
hello
hello.c
hello.gdb
hello.o
/mnt/nfs>
```

编译后生成hello可执行文件

图 13-10

执行结果如下，输出“Hello World“，注意在 hello 前要加“./”，表示执行当前目录下的程序，否则会执行/bin 下的 hello 程序。

```
/mnt/nfs> ./hello
Hello World
```

图 13-11

如果我们在 PC 主机上修改 hello 程序如下

```c
#include <stdio.h>

int main(void)
{
        printf("Hello Wold again\n\r");
        return 0;
}
```

图 13-12

然后重新编译 uClinux，这时不需要把 image 文件下载或烧写到开发板中，而是可以直接执行/mnt/nfs 下的 hello 程序，这时可以发现 hello 程序已经被更新了

```
/mnt/nfs> ./hello
Hello Wold
/mnt/nfs> ./hello
Hello Wold again
/mnt/nfs> _
```

图 13-3

从上面的例子我们可以看出，利用 NFS 可以加快我们开发应用程序的速度，碰到应用程序有问题，便在 PC 主机端进行编辑、编译，只要不重启开发板就不必做任何操作，因为 mount 的主机硬盘上的应用程序会自动覆盖更新，再重新执行的就是更改后的新版本。这样反复调试、更改、编译再调试，而不必烧写开发板，直至程序工作正常。等调试结束后，再将程序烧写到开发板中。

第十四章　uClinux 环境下用户应用程序的开发

1、在 uCinux 中添加用户的应用程序

成功编译uClinux后，我们就需要在uClinux平台上编写自己的应用程序，在开发我们的应用之前，用户需要对Linux下的编程有一定的了解，知道Makefile的作用等，如果没有这方面的经验，可以先参考《UNIX环境高级编程》。

下面我们用一个例子来作说明：

1、cd uClinux-dist/user　　注：用户开发的程序必须全部放到user目录下，不然用户需要自己定义大量的宏，费时费力。并推荐在user目录下为你的应用程序建立一个目录。

2、mkdir hello　　为应用程序建立一个目录

3、cd hello

4、vi hello.c　　用vi程序来编写我们的应用程序,如果对vi不熟，可以查阅相关文档，vi是Linux下最常用的编缉工具

hello.c的源代码如下：

```
#include <stdio.h>

int main()
{
    printf(“Hello world!\n”);
    return 0;
}  //源程序结束
```

5、vi Makefile

编写Makefile文件，这个文件用来控制我们的编译过程

Makefile的例子如下：

```
EXEC = hello
OBJS = hello.o

all: $(EXEC)

$(EXEC): $(OBJS)
```

```
	$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

romfs:
	$(ROMFSINST) /bin/$(EXEC)

clean:
	rm -f $(EXEC) *.elf *.gdb *.o
```

注：Makefile中每行前不能有空格，缩进的地方必须用TAB键，不能用空格，为了避免出错,可以直接将user/null目录下的Makefile文件复制到当前应用程序的目录下，并根据你的需要修改头两行

EXEC = hello （这里修改为你程序生成的可执行文件的文件名，我们以hello为例）

OBJS = hello.o （这里根据你源程序的文件名进行修改，我们以源程序文件名为hello.c为例，如果你的源程序文件名不是hello.c，假如为myhello.c，则将该行修改为 OBJS = myhello.o）

如果你的应用程序由几个文件组成，假设为hello1.c和hello2.c组成，则将第二行修改为：

OBJS = hello1.o hello2.o

6、修改user目录下的Makefile

为了将我们自己的应用程序编译到uclinux中，我们需要修改user目录下的Makefile文件。

在该Makefile文件中找到dir_y += games这一行，并在此行后添加一行代码：

dir_y += hello （hello是我们新建的目录，你可以将其修改为你自己应该程序的目录）

7、重新编译uclinux

注意：必须退出到uClinux-dist目录下进行编译，否则会出错！！！

可以按照上一部分的步骤进行编译，但如果你以前已经编译过uclinux内核，则只需依次执行下面的指令就可生成新的映像文件。

退出至uClinux-dist目录

make user_only

```
make romfs
make image
make linux
make image
```

2、在 make menuconfig 中加入用户应用程序的选项

当应用程序调试通过后，作为二次开发的产品，为了便于其它用户的使用，我们希望能在make menuconfig中添加我们应用程序的选项，当用户选中我们的应用程序时，就把它编译进uclinux，否则就不需要编译。为了实现这一编译开关，我们需要修改两个文件。

1、修改uClinux-dist/config目录下的config.in文件。

在这个文件的最后面添加以下代码：

```
mainmenu_option next_comment
comment ‘My Applicatons’

bool ‘Hello World’          CONFIG_USER_HELLO_WORLD
endmenu
```

2、修改uClinux-dist/user目录下的Makefile文件

在上文的描述中，我们曾经在这个文件中加入了一行代码

```
dir_y += hello
```

这行代码的目的是不管用户是否选择了这个应用程序，它都会被编译到uclinux中，为了实现让用户自由选择的目的，我们将该行代码改成

```
dir_CONFIG_USER_HELLO_WORLD += hello
```

这样修改后，仅当用户在make menuconfig中选择了Hello World，这个应用程序才会编译。

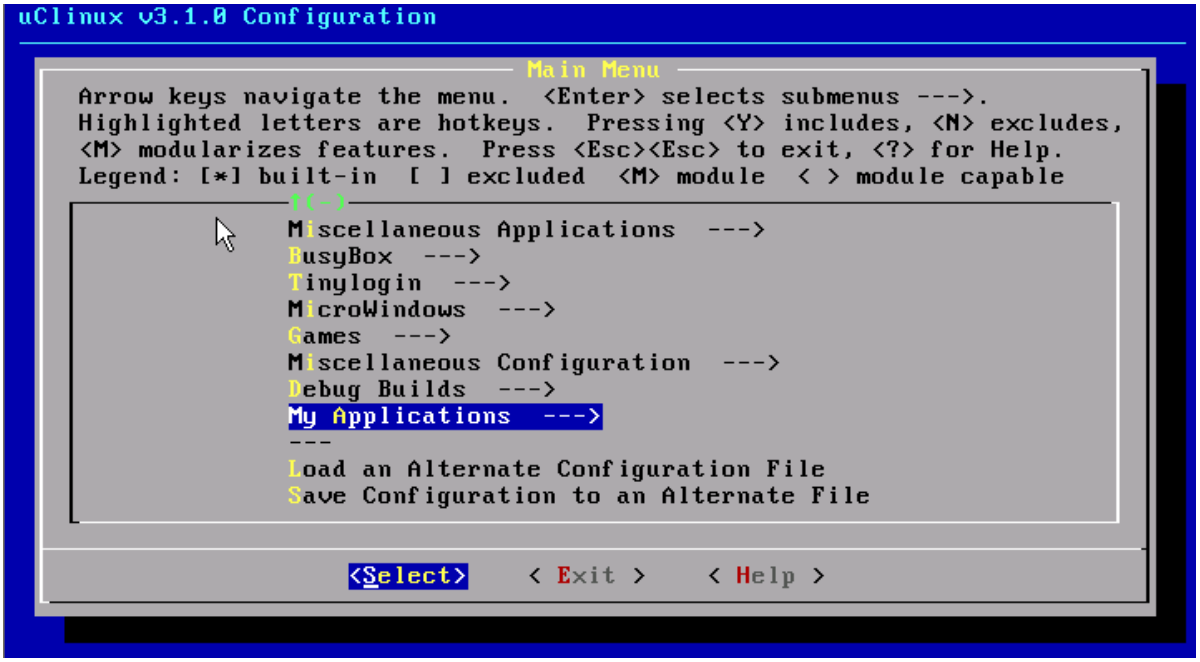


图 14-1

上图中我们看到编译配置选项中多了一项My Applications，进入该项后，如下图所示：

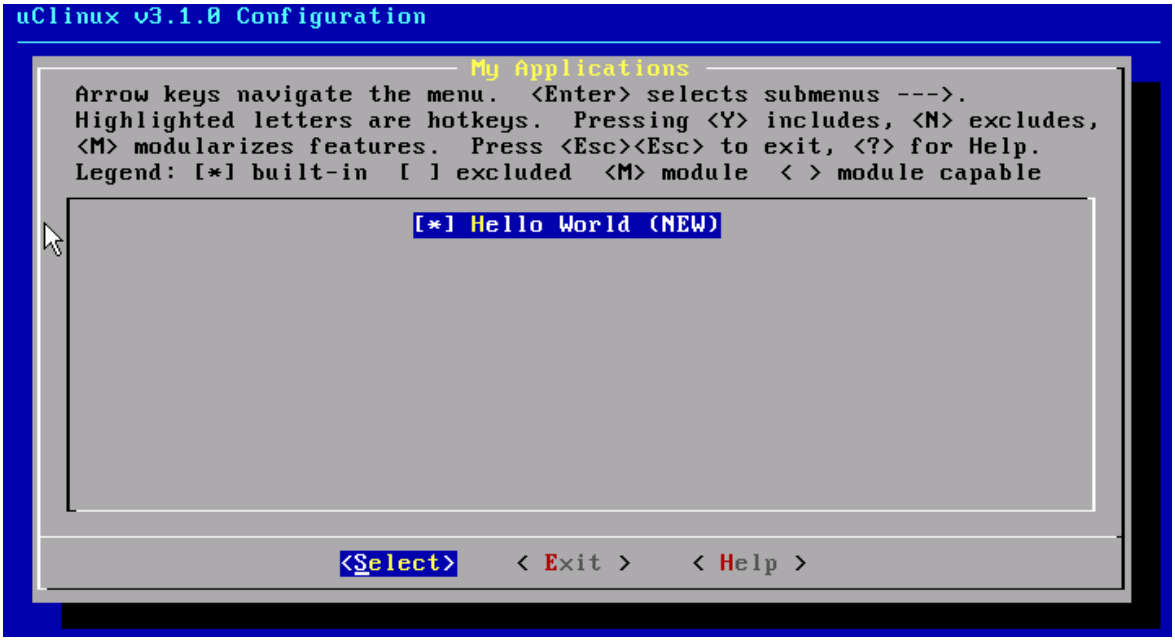


图 14-2

当我们选中Hello World的时候，我们新增的应用程序将被编译。

配制完成后，可以按正常的编译步骤，重新编译uClinux即可。

第十五章 利用 GDB 和 GDBServer 远程调试 uClinux 下的应用程序

1、GDB 简介

GDB 是 GNU C 自带的调试工具。它可以使用程序的开发者了解到程序在运行时的详细细节，从而能够很好的除去程序的错误，达到调试的目的。

使用 GDB 可以完成下面这些任务：

- 运行程序. 可以给程序加上你所需要的任何条件；
- 在规定的条件下让程序停止；
- 显示在你的程序停止的时候程序的状态；
- 在你的程序中改变据，使得你可以更好的改正程序的错误；

GDB 提供了大量的命令，用来完成程序的调试。在 GDB 的基础上还实现有 X-Windows 下的调试界面 DDD，调试界面友好。如果需要了解更多的 GDB 调试指令，请参阅相应的 GNU 文档

2、GDB 的远程调试（Remote Debugging）

在 GDB 里面有一个调试目标（Target）的概念。调试目标就是被调试程序所获得的执行环境。一般的情况下，被调试的程序和 GDB 所运行的环境是完全一样的，那么可以使用 GDB 的“file”或者“core”命令可以指定调试目标. （file 命令用来指定用来调试的可执行文件,core 命令用来指定调试的时候需要导入的 core dump 文件）。如果需要调试的程序和 GDB 所运行的环境不同，或者说需要调试的环境上根本无法运行起 GDB（如我们的开发板），那么就没有办法使用“file”或者” core”命令来指定调试目标了。这时就需要使用 GDB 的远程调试功能，通过一台可以运行 GDB 的主机，用 GDB 远程通信协议和被调试程序所在的目标平台（开发板）连接，从而调试程序。GDB 利用“target remote”命令来建立主机与目标平台的连接，启动远程调试功能，其形式如下：

target remote /dev/ttyS0　　表示 GDB 通过主机的串口 1 与开发板通信

target remote 192.168.1.100:3000 表示 GDB 通过主机的网络接口与开发板（IP 地址为 192.168.1.100）的 3000 端口建立 TCP 连接进行调试。

GDB 远程调试功能允许设计人员一步一步地调试程序代码、设置断点、查看内存变量。GDB 与开发板之间交换信息的能力很强，功能甚至相当于某些低端的 ICE 仿真器。

3、GDBServer 的简介

GDBServer 是一个基于 Linux 系统上，用于实现 GDB 远程调试功能的调试代理程序，运行在 HOST 上的 GDB 能够较为方便地通过 GDBSERVER 对嵌入式系统（开发板）的应用程序进行调试。

4、GDBServer 的工作流程

GDBServer 作为运行在开发板的一个应用程序来工作，当开发板启动后，用户通过向开发板输入命令启动。用户在命令中必须告知调试程序时的三个要素：

- GDBServer 与主机上运行的 GDB 的通信方式（串口或 LAN）
- 被调试应用程序的路径名
- 被调试应用程序的参数(如没有，则可忽略)

命令的具体形式如下：

```
gdbserver  COMM   PROGRAM   [ARGS…]
```

其中 COMM 指明 GDBServer 与 GDB 的通信方式，PROGRAM 指明被调试应用程序的路径名，ARGS 指明被调试应用的参数。如：

```
gdbserver   /dev/ttyS0    /bin/test 10
```

表示 GDBServer 通过 COM1 与主机上的 GDB 通信，被调试的应用程序为以 10 为参数启动的 test。又如：

```
gdbserver   :3000      /bin/test 10
```

表示 GDBServer 通过开发板上的网络接口使用端口 3000 与主机上的 GDB 建立 TCP 连接，并进行调试。上述指令中省略了开发板的 IP 地址 192.168.1.100，这符合 GDBServer 的语法要求，默认为开发板的 IP。

当开发板上的 GDBServer 成功启动后，用户在主机上启动 GDB，并使用 GDB 的“target remote”命令与开发板的 GDBServer 建立连接，然后就可以使用 GDB 的各

项命令。下图形象地说明了远程调试的步骤

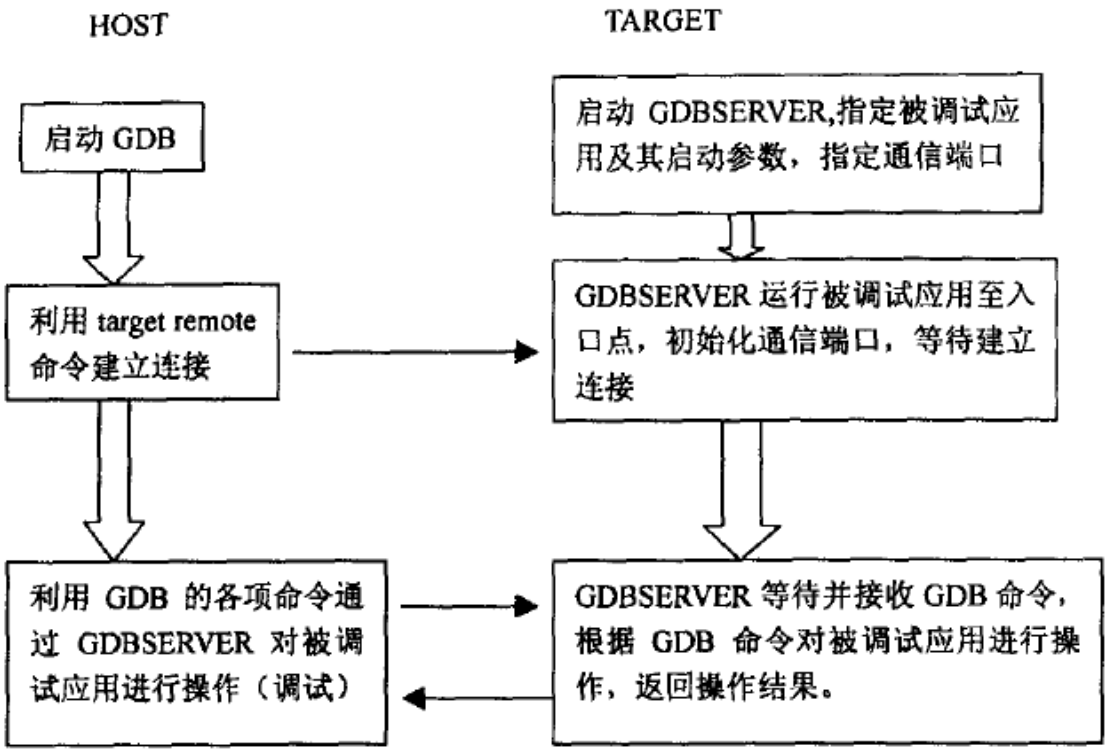


图 15-1

3、远程调试实战

我们用前面的 hello 程序来举例说明如何进行 GDB 远程调试

在开发板端执行 gdbserver，我们以通过网络接口来进行调试，命令如下：

```
gdbserver  :3000  /bin/hello
```

超级终端的提示信息如下，说明 gdbserver 已经在开发板上成功执行，正在等待 PC 主机发起连接。

```
Sash command shell (version 1.1.1)
/> gdbserver :3000 /bin/hello
Process /bin/hello created; pid = 31
```

图 15-2

在 PC 主机上输入下面的调试指令，其中 hello.gdb 包含了 hello 可执行文件的调试信息：

arm-elf-gdb hello.gdb

图 15-3

在(gdb)命令提示符下输入连接开发板的命令，其中 192.168.1.100 是开发板的 IP 地址：

target remote 192.168.1.100:3000

连接连接后，GDB 的提示信息如下

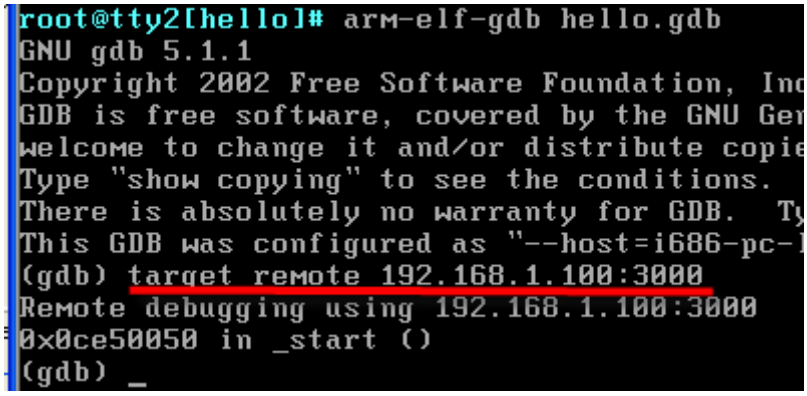


图 15-4

这时我们就可以进行调试了，在(gdb)提示符下输入调试指令

b main

上述指令表示在 main 函数的入口处设置一个断点，然后再执行

c

上述指令 c (continue)表示继续执行，程序会停在 main()函数处，指令 l 用来查看源代码。再使用单步指令 n 来跟踪程序的执行，调试过程如下图所示，红线部分为输入的指令：

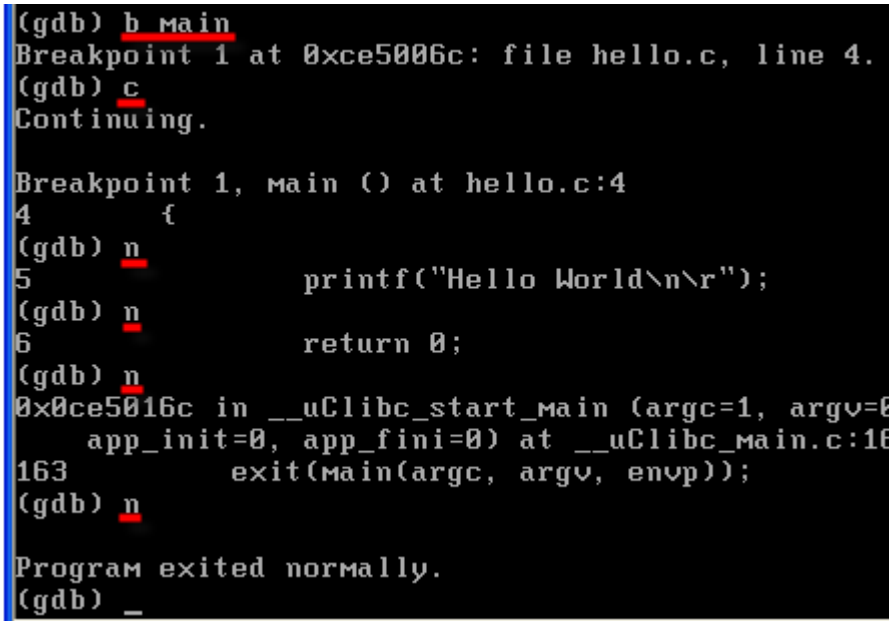


图 15-5

程序结束后，在超级终端可以看到程序的输出：

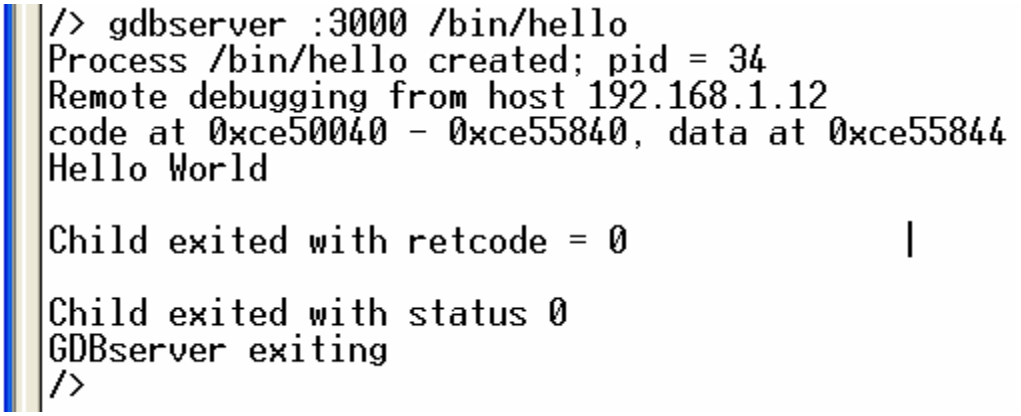


图 15-6

GDB 是一个功能十分强大的调试工具，为了开发应用程序的开发，强烈建议大家仔细研谈一下 GDB 的相关文档，熟练掌握 GDB 的调试方法，以加快程序的调试速度。

第十六章　Linux 设备驱动程序的设计开发

设备驱动在操作系统中的地位是无需多加说明的，计算机最基本的三个物质基础就是 CPU，内存以及输入/输出设备。严格的说，离开了对设备的操作，即输入/输出操作，计算机本身也就失去了意义。在 Linux 不断流行的今天，世界各地都有人在钻研 Linux 内核，大多数是在写设备驱动程序。尽管每个驱动程序都不一样，而且你还要知道自己设备的特殊性，但是这些设备驱动程序的设计原理、基本技术技巧都是一样的。

本章首先介绍了 Linux 操作系统的结构，包括 Linux 进程的管理，内存的管理，文件系统以及 Linux 内核的系统调用。之后介绍了 Linux 设备驱动程序设计原理以及在开发过程中需要注意的一些问题。

uClinux 与 Linux 在驱动开发上，大体结构只有细微的差别，因此，在本章中主要以 Linux 为基础进行说明，并说明 uClinux 和 Linux 之间的一些细小差别。

1、Linux(uClinux)操作系统的内核入门知识

Linux 设备驱动程序运行在内核空间，在加载的同时就成为内核的一部分，我们在编写 Linux 设备驱动程序的时候,难免会碰到许多与 Linux 内核相关的一些问题，所以在介绍 Linux 设备驱动程序设计之前我们有必要先介绍一下 Linux 操作系统。由于本说明书主要是介绍嵌入式 Linux 设备驱动程序的设计开发，因此这一小节将操作系统分析的重点放在与嵌入式系统有关的若干问题上。

1)Linux 和 uClinux 内核的进程管理

进程管理在内核中负责创建和终止进程，并且处理它们和外部世界的联系（输入和输出)。不同进程之间的通信(信号，管道，消息队列，共享内存)是基本的功能，这也是由内核来实现的。另外，调度器可能是整个操作系统中最关键的进程，这也是进程管理中的一部分。更广义的说，内核的进程管理实现了一个 CPU 上实现多个进程的抽象概念。理解嵌入式系统的进程管理，对于在该系统中创建进程、进行多线程编程是十分有利的。实际上不同的嵌入式系统，他们的进程管理机制大相径庭，

例如:它们允许同时创建进程的个数就不相同。

在多任务的操作系统环境下，各个程序是并发执行的，它们共享系统资源，共同决定这些资源的状态。彼此间相互制约、相互依赖，因而呈现出并发、动态及互相制约等新的特征。这样，用程序这个静态概念已不能如实反映程序活动的这些特征。为此，人们引进了进程（Process）这一新概念，来描述程序动态执行过程的性质。

简单说来，进程就是程序的一次执行过程。它有着走走停停的活动规律。进程的动态性质是由其状态变化决定的。通常在操作系统中，进程至少要有三种基本状态，它们是运行态、就绪态和封锁态（或阻塞态）。而在Linux中，进程有如下几种运行状态：

- 运行状态(TASK_RUNNING)：是指当前进程已分配到CPU，它的程序正在处理器上执行时的状态。处于这种状态的进程个数不能大于CPU的数目。在一般单CPU体系结构中，任何时刻处于运行状态的进程至多有一个。
- 就绪状态：指进程已具备运行条件，但因为其它进程正占用CPU，所以暂时不能运行而等待分配CPU的状态。一旦把CPU分给它，立即就可运行。在操作系统中，处于就绪状态的进程数目可以是多个。
- 等待状态：进程因等待某种事件发生（例如等待某一输入、输出操作完成，等待其它进程发来的信号等）而暂时不能运行的状态。也就是说，处于这一状态的进程尚不具备运行条件，即使CPU空闲，它也无法使用。这种状态有时也称为不可运行状态或挂起状态，系统中处于这种状态的进程可以是多个的。Linux系统分为两种等待进程：可中断的(TASK_INTERRUPTIBLE)和不可中断的(TASK_UNINTERRUPTIBLE)。可中断的等待进程可以被某一信号(Signal)中断；而不可中断的等待进程不受信号的打扰，将一直等待硬件状态的改变。
- 停止态(TASK_STOPPED) 进程被停止，通常是通过接收一个信号。正在被调试的进程可能处于停止状态。
- 僵死态(TASK_ZOMBIE) 由于某些原因被终止的进程，但是该进程的控制结构task_struct仍然保留着。

在程序调试中，得到 CPU 运行的进程是被称为 current 进程，但是该进程会由于

申请资源、进行 I/O 操作、收到信号等多种原因，而不得不放弃 CPU 而处于等待状态。此时调度进程会根据优先权等因素选择合适的进程分配 CPU。处于等待状态的进程，会继续等待条件的满足(如获得申请的资源，收到信号等等)， 一旦条件满足，则进程状态将变为 TASK_RUNNING 并被挂入运行队列，从而有机会分配到 CPU 继续执行下去。处于运行状态的进程收到跟踪调试信号(SIGSTOP)或者调用跟踪的系统调用函数，会通过设置标志位 PF_PTRACED 或 PF_TRACESYS 信号，将进程的状态改变为 TASK_STOPPED 状态。

处于暂停状态的进程在收到 SIGKILL 或 SIGCONT 信号之后，会被唤醒，进程的状态将改变为 TASK_RUNNING。最终进程的结束是通过调用 do_exit 释放所占用的资源(除 PCB 结构之外)，并且该进程的状态切换到 TASK_ZOMBIE 并等待 sys_wait4 释放处于 TASK_ZOMBIE 状态的子进程的 PCB(Process Control Block 进程控制块)。

进程状态切换如下图所示：

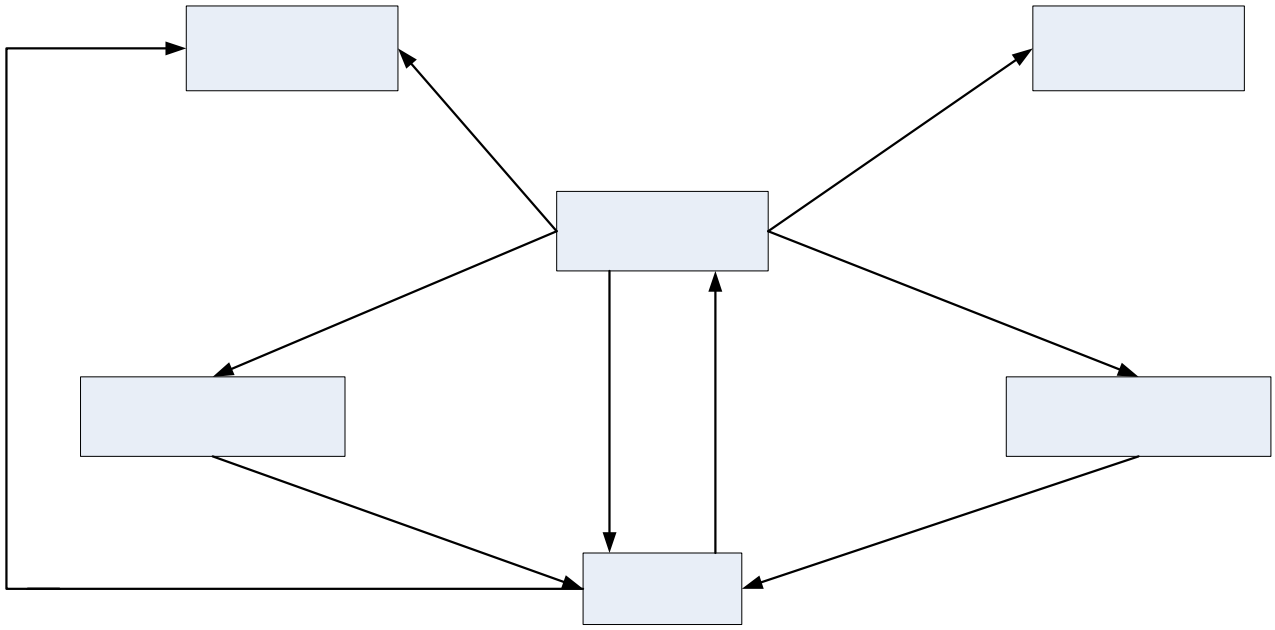


图 16-1

2)Linux 和 uClinux 的内存管理

内存管理是操作系统的核心，它负责管理计算机系统的存储器。因为 ARM7 处理

器是没有 MMU(Memmory Manage Unit 内存管理单元)，因此 Linux 和 uClinux 在内存管理方面存在比较大的差异。

- **标准 Linux 使用的虚拟存储器技术**

标准Linux 使用虚拟存储器技术，这种技术用于提供比计算机系统中实际使用的物理内存大得多的内存空间。使用者将感觉到好像程序可以使用非常大的内存空间，从而使得编程人员在写程序时不用考虑计算机中的物理内存的实际容量。为了支持虚拟存储管理器的管理，linux 系统采用分页（paging）的方式来载入进程。所谓分页既是把实际的存储器分割为相同大小的段，例如每个段1024 个字节，这样1024 个字节大小的段便称为一个页面（page）。虚拟存储器由存储器管理机制及一个大容量的快速硬盘存储器支持。它的实现基于局部性原理，当一个程序在运行之前，没有必要全部装入内存，而是仅将那些当前要运行的那些部分页面或段装入内存运行(copy-onwrite)，其余暂时留在硬盘上程序运行时如果它所要访问的页(段)已存在，则程序继续运行，如果发现不存在的页（段），操作系统将产生一个页错误（page fault），这个错误导致操作系统把需要运行的部分加载到内存中。必要时操作系统还可以把不需要的内存页（段）交换到磁盘上。利用这样的方式管理存储器，便可把一个进程所需要用到的存储器以化整为零的方式，按需求分批载入，而核心程序则凭借属于每个页面的页码来完成寻址各个存储器区段的工作。

标准Linux 是针对有内存管理单元的处理器设计的。在这种处理器上，虚拟地址被送到内存管理单元（MMU），把虚拟地址映射为物理地址。通过赋予每个任务不同的虚拟——物理地址转换映射，支持不同任务之间的保护。地址转换函数在每一个任务中定义，在一个任务中的虚拟地址空间映射到物理内存的一个部分，而另一个任务的虚拟地址空间映射到物理存储器中的另外区域。计算机的存储管理单元（MMU）一般有一组寄存器来标识当前运行的进程的转换表。在当前进程将CPU放弃给另一个进程时（一次上下文切换），内核通过指向新进程地址转换表的指针加载这些寄存器。MMU 寄存器是有特权的，只能在内核态才能访问。这就保证了一个进程只能访问自己用户空间内的地址，而不会访问和修改其它进程的空间。当可执行文件被加载时，加载器根据缺省的ld 文件，把程序加载到虚拟内存的一个空间，因

为这个原因实际上很多程序的虚拟地址空间是相同的，但是由于转换函数不同，所以实际所处的内存区域也不同。而对于多进程管理，当处理器进行进程切换并执行一个新任务时，一个重要部分就是为新任务切换任务转换表。我们可以看到Linux系统的内存管理至少实现了以下功能：

运行比内存还要大的程序。理想情况下应该可以运行任意大小的程序

可以运行只加载了部分的程序，缩短了程序启动的时间

可以使多个程序同时驻留在内存中提高CPU的利用率

可以运行重定位程序。即程序可以方于内存中的任何一处，而且可以在执行过程中移动。

编写机器无关的代码。程序不必事先约定机器的配置情况。

减轻程序员分配和管理内存资源的负担。

可以进行共享——例如，如果两个进程运行同一个程序，它们应该可以共享程序代码的同一个副本。

提供内存保护，进程不能以非授权方式访问或修改页面，内核保护单个进程的数据和代码以防止其它进程修改它们。否则，用户程序可能会偶然（或恶意）的破坏内核或其它用户程序。

虚存系统并不是没有代价的。内存管理需要地址转换表和其他一些数据结构，留给程序的内存减少了。地址转换增加了每一条指令的执行时间，而对于有额外内存操作的指令会更严重。当进程访问不在内存的页面时，系统发生失效。系统处理该失效，并将页面加载到内存中，这需要极耗时间的磁盘I/O 操作。总之内存管理活动占用了相当一部分CPU时间（在较忙的系统中大约占10％）。

- **uClinux 针对 NOMMU 的特殊处理**

对于uClinux来说，其设计针对没有MMU的处理器，即uClinux不能使用处理器的虚拟内存管理技术（应该说这种不带有MMU的处理器在嵌入式设备中相当普遍）。uClinux仍然采用存储器的分页管理，系统在启动时把实际存储器进行分页。在加载应用程序时程序分页加载。但是由于没有MMU管理，所以实际上uClinux采用实存储器管理策略（real memory management）。这一点影响了系统工作的很多方面。

uClinux系统对于内存的访问是直接的，（它对地址的访问不需要经过MMU，而是直接送到地址线上输出），所有程序中访问的地址都是实际的物理地址。操作系统对内存空间没有保护（这实际上是很多嵌入式系统的特点），各个进程实际上共享一个运行空间（没有独立的地址转换表）。

一个进程在执行前，系统必须为进程分配足够的连续地址空间，然后全部载入主存储器的连续空间中。与之相对应的是标准Linux系统在分配内存时没有必要保证实际物理存储空间是连续的，而只要保证虚存地址空间连续就可以了。另外一个方面程序加载地址与预期（ld文件中指出的）通常都不相同，这样relocation过程就是必须的。此外磁盘交换空间也是无法使用的，系统执行时如果缺少内存将无法通过磁盘交换来得到改善。uClinux对内存的管理减少同时就给开发人员提出了更高的要求。如果从易用性这一点来说，uClinux 的内存管理是一种倒退，退回了到了UNIX早期或是Dos系统时代。开发人员不得不参与系统的内存管理。从编译内核开始，开发人员必须告诉系统这块开发板到底拥有多少的内存（假如你欺骗了系统，那将在后面运行程序时受到惩罚），从而系统将在启动的初始化阶段对内存进行分页，并且标记已使用的和未使用的内存。系统将在运行应用时使用这些分页内存。从内存的访问角度来看，开发人员的权利增大了（开发人员在编程时可以访问任意的地址空间），但与此同时系统的安全性也大为下降。此外，系统对多进程的管理将有很大的变化，详见下一小节。

3)在无 MMU 的基础上 uClinux 多进程的处理

uClinux 不支持 MMU，在实现多个进程时（fork 调用生成子进程）需要实现数据保护。

由于没有 MMU 的支持，不能实现 COW(Copy-On-Write)技术，而 fork()的实现需要采用 COW 技术，因此在 uClibc 库中没有提供 fork()，即我们在 uClinux 应用的代码中不能使用 fork()。uClinux 的开发者建议使用 vfork()来代替 fork()。

fork()和 vfork()存在以下差异：

在调用完 fork()后，子进程将获得父进程的堆栈，数据段，用户空间和 PCB(Process Control 进 Block 进程控制块)的一个 copy。但是由于经常在调用

fork()后，紧跟着调用 exec，为了节约系统资源，在实现 fork()时并不做一个父进程的堆栈、数据段，用户空间的 copy，而是使这些区域由父、子进程共享，而且由内核将这些区域的存取权限改为只读。如果有进程试图写这些区域，则内存为相关部分（如虚拟内存中的某一页）做一个 copy，这就是所谓的 COW 技术，也就是指在必要的时候才进行数据复制的工作，以节省系统运行资源。调用 fork()后，父、子进程中的某一个修改共有的变量，关闭共有的文件描述符时，对另一进程不会造成影响。

在调用完 vfork()后，vfork()创建的子进程在调用 exec 或 exit 前，将与父进程共享地址空间，即如果子进程中对某一变量进行了建立、修改、释放等操作，等效于父进程对该变量也做了相应的操作。vfork()不同于 fork()的第二点还在于 vfork()的实现中总是保证子进程优先运行，在子进程调用了 exec 或 exit 后，父进程才能被调度运行。（注意：如果在调用 exec 或 exit 前，子进程依赖于父进程的进一步动作，将会造成死锁！）

4) Linux 内核的文件系统

操作系统中有两个最重要的部分，一个就是进程调度管理，另一个就是文件系统了。事实上，有些操作系统(如一些嵌入式操作系统)可能有进程管理而没有文件系统;而有些操作系统(如 MSDOS)则只有文件系统而没有进程管理。但是如果两者都没有则就不能称其为“操作系统”了。

Linux 操作系统不仅支持多个不同的文件系统如 EXT2,MSDOS,VFAT 等还支持这些文件系统相互间的访问。每种文件系统都有自己的组织结构和文件操作函数，相互之间差别很大。Linux 对于上述文件系统的支持是通过虚拟文件系统(VFS)的引入而实现的，如下图所示：

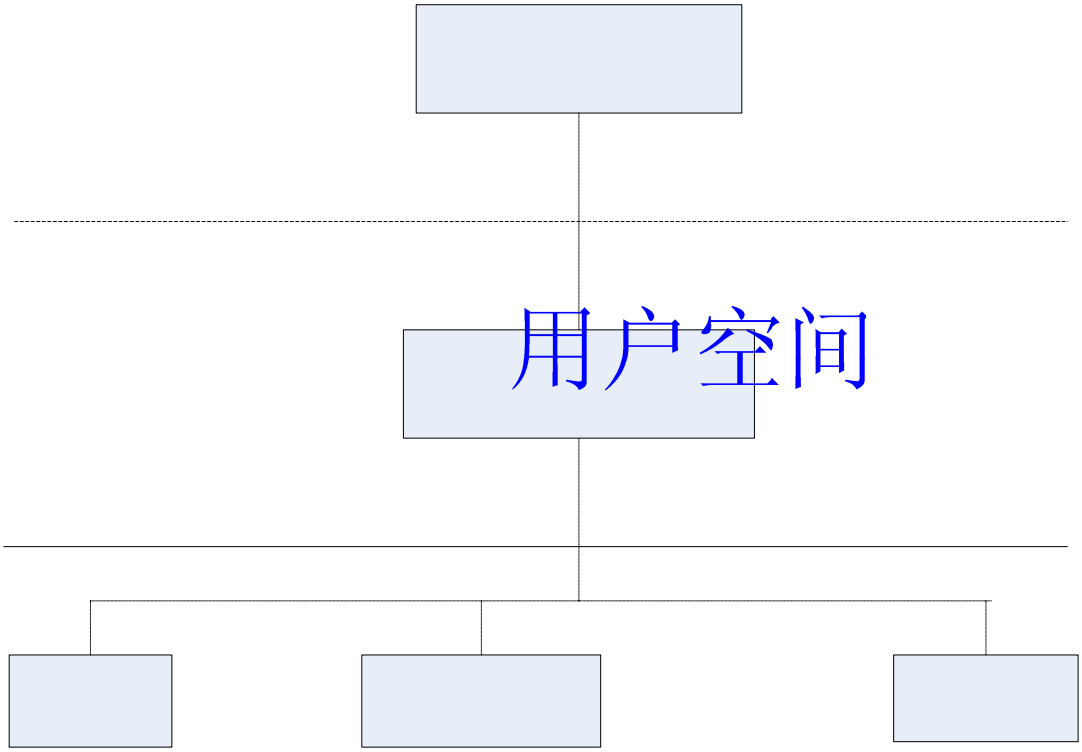


图 16-2

虚拟文件系统的所有数据结构都是在系统运行之后才建立的，在系统卸载的时候再删除。磁盘上没有虚拟文件系统的数据结构，它并不是一个真正的文件系统。要想让系统工作,必须要有像 ext2 这样的逻辑文件系统并且让虚拟文件系统和逻辑文件系统之间建立逻辑连接。VFS 为各个逻辑文件系统与 Linux 内核进行通信提供了一致的接口，用户想要在 Linux 中加入自己开发的文件系统就必须按照这个接口的要求来编写文件系统的操作函数。实际上在这种情况下，代表新开发的文件系统与内核打交道的是 VFS。

VFS 是逻辑文件系统与服务之间的一个接口层,它对 Linux 的每个文件系统的所有细节进行抽象，使得不同的文件系统在 Linux 核心以及系统中运行的其他进程来看，都是相同的。它只存在于内存中，不存在于任何外存空间。总之，VFS 的功能包括:

(1)记录可用的文件系统的类型；

(2)将设备与对应的文件系统联系起来；

(3)处理一些面向文件的通用操作；

(4)涉及到针对文件系统的操作时，VFS 把它们映射到与控制文件、目录、以及

inode 相关的逻辑文件系统。

2、Linux(uClinux)设备驱动程序的入门知识

1）Linux 设备驱动程序的概念

从程序结构角度而言，驱动程序是子程序和数据的集合，是输入输出设备的软件接口，它的任务是向系统提供接口函数，所以简单的说编写驱动程序就是实现这些接口函数。

设备驱动程序是一组由内核中的相关子例程进行特殊的操作时，它就调用相应的驱动例程。这就使得控制从用户进程转移到了驱动例程，当驱动例程完成后控制又被返回至用户进行。其层次关系见下面：

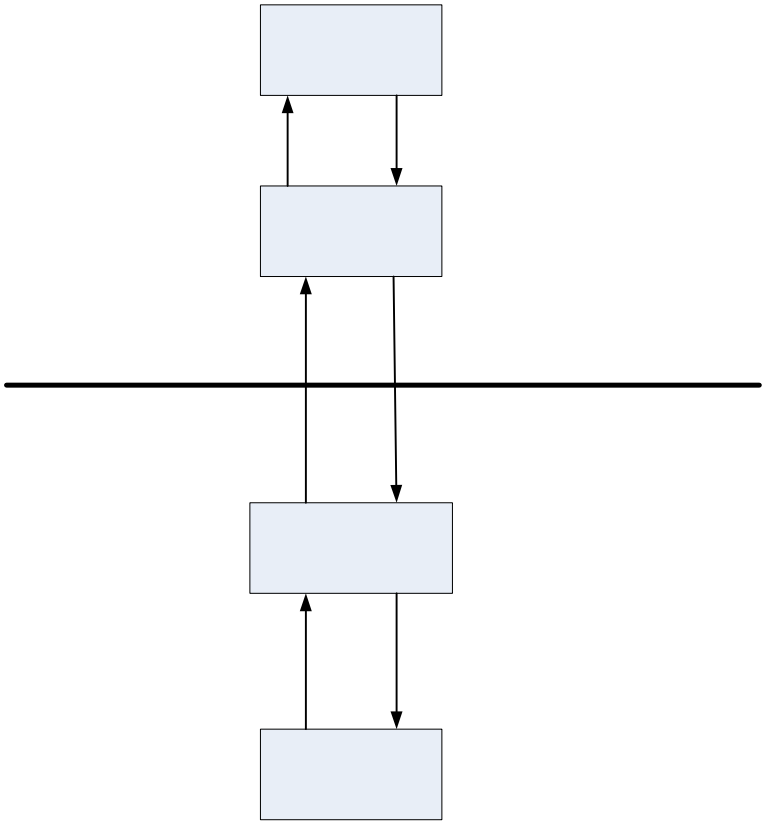


图 16-3

从上图可以看出，系统调用的是操作系统内核、应用程序之间的接口，设备驱动程序是操作系统内核、机器硬件之间的接口。设备驱动程序为应用程序屏蔽了硬件的细节，这样在应用程序看来，硬件设备只是一个设备文件，应用程序可以像

操作普通文件一样对硬件设备进行操作。设备驱动程序是内核的一部分，它完成以下功能：

- 对设备初始化和释放
- 把数据从内核传送到硬件和从硬件中读取数据
- 读取应用程序传送给设备文件的数据和回送应用程序请求的数据。
- 检测和处理设备中出现的错误。

每个设备驱动程序都具有以下几个特性

- 具有一整套和硬件设备通讯的例程,并且提供给操作系统一套标准的软件接口；
- 具有一个可以被操作系统动态地调用和移除的自包含组件；
- 可以控制和管理用户程序和物理设备之间的数据流；

由于存在着这些特点，我们可以把它们的共同点提炼出来，形成一个驱动程序的框架，在 Linux 中以某些特殊的数据结构来体现，详见下文。下图为设备驱动程序与操作系统和设备之间的关系。

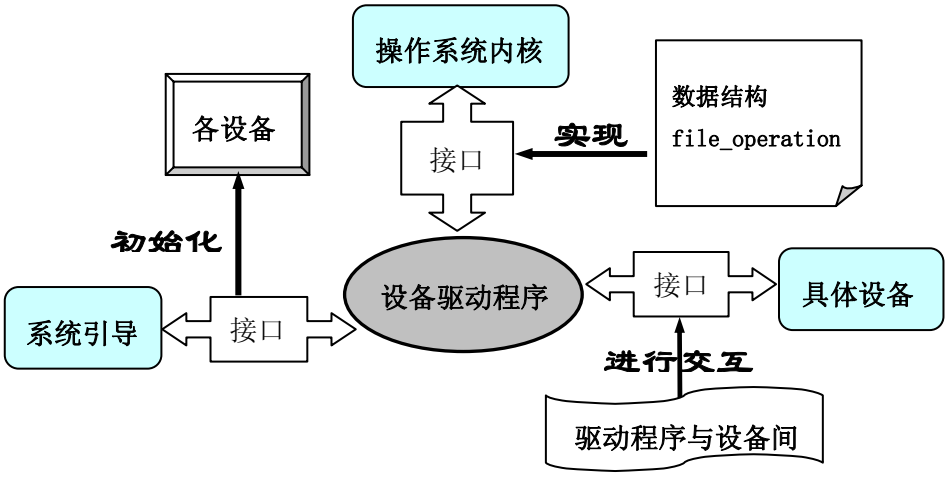


图　16-4

2）Linux 设备驱动程序的分类

Linux 将设备按照功能特性划分为三种类型：字符设备，块设备，网络设备。网络设备是一种特殊的设备，但是由于其处理数据也是块存块取的，所以也有人将其

划为块设备，但是其访问机制与块设备还是有很大的不同，所以我们还是将其做为一个独立的设备类型进行介绍。

字符设备

字符设备可以象访问文件一样访问字符设备,字符设备驱动程序负责实现这些行为这样的驱动程序通常会实现 open, close, read 和 write 等系统调用。系统控制台和并口就是字符设备的例子，它们可以很好的用流的概念来加以描述。通过文件系统节点可以对字符设备加以访问。字符设备和普通文件系统的唯一区别是:普通文件允许在其上来回读写，而大多数字符设备仅仅是数据通道，只能顺序读写（如串口）。当然也存在这样的设备，看起来象一个数据缓冲区，可以来回读写数据。

块设备

块设备是文件系统的宿主，如硬盘。在大多数 Unix 系统中，只能将块设备当作多个块进行访问，一块通常是 lkb 数据。而 Linux 允许你象字符设备那样读取块设备－－即允许一次传输任意长度的字节。结果是，块设备和字符设备只在内核内部的管理上有所区别，因此也就是在内核/驱动程序间的软件接口上有所区别。就象字符设备一样，每个块设备也通过文件系统节点来读写数据，它们之间的不同对于用户来说是透明的。块设备驱动程序和内核的接口和字符设备驱动程序的接口是一样的，它也通过一个传统的面向块的接口与内核通信，但这个接口对于用户来说是不可见的。

网络设备

任何网络事件的处理都是通过接口实现的。通常，接口是一个硬件设备，但也可以象 loopback，回路)接口一样，是软件设备。网络接口是由内核网络子系统驱动的，它负责发送和接收数据包，而且无需了解每次事务是如何映射到实际被发送的数据包。由于不是面向流的设备，网络接口不能象/dev/ttyS0 那样简单地映射到文件系统的节点上。Linux 调用这些节点的方式是给它们分配个独立的名字如 eth0，这样的名字在文件系统中并没有对应项。内核和网络设备驱动程序之间的通信，与

字符设备和块设备驱动程序与内核的通信是完全不一样的。不再是通过 read, write 函数数而是采用 sk_buff 数据缓冲区，调用与数据包相关的函数。

3） 模块化机制

Linux 中的可加载模块(Module)是 Linux 内核支持的动态可加载模块,它们是内核的一部分（通常是设备驱动程序），但是没有编译到内核里。Module 可以单独编译成目标代码，Module 是一个目标文件，它可以根据需要在系统启动后动态地加载到系统核心中。当 Module 不再被需要时，可以动态地卸载出系统内核。Linux 中大多数设备驱动程序或文件都是 Module 模式的。超级用户可以通过 insmod 和 rmmod 命令显式地将 Module 载入内核或从内核中将它卸载。内核也可以在需要的时候，请求守护进程(kerneld)自动装载和卸载 module。通过动态将代码载入内核可以减小内核代码的规模，使内核配置更为灵活。若在调度新驱动程序时采用 Module 技术，用户不必每次修改后都需要编译内核和启动系统。

事实上，uClinux 在 ARM7 等嵌入式系统中，为了内核精简的考虑，我们通过配置内核，然后把驱动程序采用静态的方式编译到内核中去的，但驱动程序仍是一个模块，只是不需要使用 insmod 和 rmmod 命令来装载和卸载驱动程序模块。在内核启动时，会自动加载驱动程序模块，也就是调用驱动程序口的 init_module()函数。加载成功后，系统将分配给该驱动程序一个主设备号。通过这个主设备号，我们就可以建立设备文件，以建立起设备和设备驱动程序联系的桥梁。通常，我们都把这些设备文件（设备节点）放在/dev 目录下，使用 mknod 命令：

mknod /dev/设备名 -c/-b 主设备号 次设备号

其中-c 参数表示建立的是字符设备，-b 表示建立的是块设备。主设备号可以通过 cat /proc/devices 看到，次设备号由自己定，用来区别同一类型的不同设备，如硬盘的 C 分区和 D 分区的主设备号相同，次设备号不同，建立了设备以后，我们就可以通过用户程序来使用驱动程序了。删除设备文件用 rm /dev/设备文件名。建立与删除设备文件（设备节点）需要系统管理员权限。

4) 用户空间和内核空间

现代的计算机体系结构中存储管理通常都包含保护机制。提供保护的目的，是要避免系统中的一个任务访问属于另外的或属于操作系统的存储区域。如在 Intel x86 体系中，就提供了特权级这种保护机制，通过特权级别的区别来限制对存储区域的访问。 基于这种构架，Linux 操作系统对自身进行了划分：一部分核心软件独立于普通应用程序，运行在较高的特权级别上，（Linux 使用 Intel 体系的特权级 3 来运行内核）。它们驻留在被保护的内存空间上，拥有访问硬件设备的所有权限，Linux 将此称为内核空间。

相对的，其它部分被作为应用程序在用户空间执行。它们只能看到允许它们使用的部分系统资源，并且不能使用某些特定的系统功能，不能直接访问硬件，不能直接访问内核空间，当然还有其他一些具体的使用限制。（Linux 使用 Intel 体系的特权级 0 来运行用户程序。）

从安全角度讲将用户空间和内核空间置于这种非对称访问机制下是很有效的，它能抵御恶意用户的窥探，也能防止质量低劣的用户程序的侵害，从而使系统运行得更稳定可靠。但是，如果像这样完全不允许用户程序访问和使用内核空间的资源，那么我们的系统就无法提供任何有意义的功能了。为了方便用户程序使用在内核空间才能完全控制的资源，而又不违反上述的特权规定，从硬件体系结构本身到操作系统，都定义了标准的访问界面。关于 x86 系统的细节，请查阅相关资料。

一般的硬件体系机构都提供一种“门”机制。“门”的含义是指在发生了特定事件的时候低特权的应用程序可以通过这些“门”进入高特权的内核空间。对于各种计算机体系来说，Linux 操作系统正是利用了“系统门”这个硬件界面，构造了形形色色的系统调用作为软件界面，为应用程序从用户态陷入到内核态提供了通道。通过“系统调用”使用“系统门”并不需要特别的权限，但陷入到内核的具体位置却不是随意的，这个位置由“系统调用”来指定，有这样的限制才能保证内核安全。我们可以形象地描述这种机制：作为一个游客，你可以买票要求进入野生动物园，但你必须老老实实的坐在观光车上，按照规定的路线观光游览。当然，不准下车，因为那样太危险，不是让你丢掉小命，就是让你吓坏了野生动物。

出于效率和代码大小的考虑，内核程序不能使用标准库函数（当然还有其它的顾虑，详细原因请查阅相关参考资料，如在内核空间中，我们必须要用 printk 来代替 printf。）因此内核开发不如用户程序开发那么方便。而且由于目前的内核是“非抢占”的，因此正在内核空间运行的进程是不会被其他进程取代的（除非该进程主动放弃 CPU 的控制，比如调用 sleep(),schedule()等），所以无论是在进程上下文中（比如正在运行 read 系统调用），还是在中断上下文(正在中断服务程序中)，内核程序都不能长时间占用 CPU，否则其它程序将无法执行，只能等待。

5) I/O 端口访问

和硬件打交道离不开 I/O 端口，老的 ISA 设备经常是占用实际的 I/O 端口，在 Linux 下，操作系统没有对 I/O 端口进行屏蔽，也就是说，任何驱动程序都可以对任意的 I/O 端口操作，这样就很容易引起混乱。每个驱动程序应该自己避免误用端口，有两个重要的内核函数可以保证驱动程序做到这一点。

1) check_region(int io_port,int off_set)

这个函数察看系统的 I/O 表，看是否有别的驱动程序占用某一段 I/O 口。

参数 1: I/O 端口的基地址

参数 2: I/O 端口占用的范围。

返回值:0 没有占用，非 0，已经被占用。

2) request_region(int io_port,int off_set,char* devname)

如果这段 I/O 端口没有被占用，在我们的驱动程序中就可以使用它，且在使用之前，必须向系统登记，以防止被其他程序占用。登记后在/proc/ioports 文件中可以看到你登记的 I/O 口。

参数 1: I/O 端口的基地址。

参数 2: I/O 端口占用的范围。

参数 3:使用这段 io 地址的设备名。

在对 I/O 口登记后，就可以放心地用 inb(),outb()之类的函数来访问了。还有一些 PCI 设备，I/O 端口被映射到一段内存中去，要访问这些端口就相当于访问一段内存了。

6） 内存的操作

我们要经常性地获得一块内存的物理地址，在设备驱动程序中动态开辟内存不是 malloc，而是 kmalloc，或者用 get_free_pages 直接申请空闲内存页，释放内存用的是 kfree,或 free_pages。请注意,kmalloc 等函数返回的是物理地址,而 malloc 等返回的是线性地址。但在无MMU的ARM7体系中,物理地址与线性地址等价。kmalloc 最大只能开辟 128k-16 字节大小的内存空间,其中的 16 个字节是被页描述符结构占用了。

7） 设备驱动程序中的主设备号和次设备号

Linux 访问设备是通过文件系统内的设备名称进行的。那些名称被称为特殊文件、设备文件或者简单称之为文件系统树的节点，它们通常位于/dev 目录下。字符设备驱动程序的设备文件可以通过 ls -l 命令输出的第一列中的“c”来识别，而”b”则表明，这是一个块设备。

如果执行 ls -l 命令，就可以在设备文件项的最后修改日期前看到两个数（用逗号分隔），这两个数就是相应设备的主设备号和次设备号。下面的列表给出了一个系统中的设备。从中可以看出它们的主设备号分别为 10，14 和 29，而次设备号分别为 175、134、3、4、20、36、52、7 和 0。

```
crw-rw---- 1 root     video      10, 175   2005-07-08    06:38  agpgart
crw-rw---- 1 root     root       10, 134   2005-07-09    00:33  apm_bios
crw-rw---- 1 root     root       10,   3   2004-09-18    07:52  atibm
crw-rw---- 1 root     audio      14,   4   2004-09-18    07:52  audio
crw-rw---- 1 root     audio      14,  20   2004-09-18    07:52  audio1
crw-rw---- 1 root     audio      14,  36   2004-09-18    07:52  audio2
crw-rw---- 1 root     audio      14,  52   2004-09-18    07:52  audio3
crw-rw---- 1 root     audio      14,   7   2004-09-18    07:52 audioioctl
brw-rw---- 1 root     cdrom      29,   0   2004-09-18    07:53  aztcd0
```

主设备号标识设备对应的驱动程序。例如/dev/video 由驱动程序 10 管理，而音频设备/dev/audio 则由驱动程序 14 管理。内核利用主设备号在 open 操作中将设备与相应的驱动程序对应起来。

次设备号只是由那些主设备号已经确定的驱动程序使用，内核的其他部分不会用到它，它只是把它传递给驱动程序。一个驱动程序管理多个设备是常用的事情，

如上表中的/dev/audio 设备，它由驱动程序 10 管理，并且该驱动程序管理着次设备号为 4、20、46、52 和 7 的数个设备。在驱动程序内部，是通过次设备号来区分不同设备的。

主设备号是用来索引设备静态数组的一个小整数，现在的 uClinux 多采用 2.4 的内核，可以支持 256 种设备（从 0 到 255，但 0 和 255 设为保留值，不能被使用）。次设备号同样为 0 到 255。

通过/dev 目录下的设备文件，我们可以将一个设备与系统内核中的驱动程序联系起来。一但驱动程序被注册到内核中，它的操作就和指定的主设备号对应了起来。当我们在与主设备号对应的设备文件上进行某个操作时，内核将从 file_operations 结构中（该结构将在下一小节中详细介绍）找到并调用正确的函数。

当我们的驱动程序编写后，需要/dev 目录下建立一个设备文件？如上文所述，在文件系统上创建一个设备节点的命令是 mknod，只有超级用户(root 权限)方可创建设备节点，除了要创建的节点名字外，该命令还带有三个参数。如：

```
mknod   /dev/test  c    253      0
```

将会创建一个字符设备(c)，主设备号为 253，次设备号为 0。在嵌入式系统的 uClinux 中，由于采用的是 romfs 文件系统，它是只读的文件系统，该文件系统一但下载到开发板中，就无法更改，也就是说不能实时建立和删除/dev 目录下的设备节点。通常是通过修改相应的 Makefile 文件来实现建立和删除设备节点的功能。我们会在下文中介绍如何修改该 Makefile 文件

8） Linux 设备驱动程序的主要数据结构

- struct file_operatoins

操作系统将每个外部设备当作文件来处理，内核通过 file_operations 结构来访问驱动程序的功能。file_operations 的定义在文件<linux/fs.h>中。

每个字符设备都有一个 file_operations 结构。这个结构指向一组操作函数(open,read…)。每个函数的定义由驱动程序提供。当然，有些标准操作某些设备并不支持，我们只需要实现该设备支持的操作就可以了。事实上，驱动程序的编写主

要就是完成这个结构里的那些函数的，如 read,write,ioctl 等。随着 linux 内核的不断升级，file_operations 结构也不断变大。最新的版本中，甚至有些函数原型都发生了一些变化。当然，新版本总会向下兼容的。

以下是 Linux 内核中 file_operations 结构的定义：

```
struct file_operations {
loff_t (*llseek) (struct file *, loff_t, int);
```

方法 llseek 用来修改文件的当前的读写位置，并将新位置作为（正的）返回值返回。参数 loff_t 是一个“长偏移量”，即使在 32 位平台上也至少占用至少 64 位的数据宽度。出错时返回一个负的返回值。如果驱动程序没有设置这个函数，那么相对于文件末尾（end-of-file，EOF）的定位操作就会失败，而其它的定位操作都将修改 file 结构（在下面会介绍 file 数据结构）中的位置计数器，并成功返回。

```
ssize_t (*read) (struct file *, char *, size_t, loff_t *);
```

用来从设备中读取数据。当该函数指针被赋为 NULL 值时，将导致 read 系统调用出错并返回-EINVAL(“Invalid Argument,非法参数”)。函数返回非负值表示成功读取的字节数。

```
ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
```

用于向设备发送数据。如果没有这个函数，write 系统调用会向调用程序返回一个－EINVAL。如果返回值非负，则表示成功写入的字节数。

```
int (*readdir) (struct file *, void *, filldir_t);
```

对于设备文件来说，这个字段应该为 NULL。它仅用于读取目录，并且只对文件系统有用。

```
unsigned int (*poll) (struct file *, struct poll_table_struct *);
```

poll 方法是 poll 和 select 这两个系统调用的后端实现。这两个系统调用可用来查询设备是否可读或可写，或是否处于某种特殊状态。这两个系统调用是可阻塞的，直至设备变为可读或可写状态为止。如果驱动程序没有定义它的 poll 方法，它所驱动的设备就会认为既可读也可写。并且不会处于其它的特殊状态。返回值是一个描述设备状态的位掩码。

```
int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
```

系统调用 iotcl 提供了一种执行设备特定的命令的方法（如格式化软盘的某个磁盘，这既不是读操作也不是写操作）。另外内核还能识别一部分 ioctl 命令，而不必调用 fops 表中的 ioctl。如果设备不提供 ioctl 入口函数，则对于任何内核未预先定义的请求，ioctl 系统调用将返回错误（-ENOTTY, “No such ioctl for device，该设备无此 ioctl 命令”）。如果该设备方法返回一个非负值，那么相同的值会被返回给调用程序以表示调用成功。

```
int (*mmap) (struct file *, struct file *);
```

mmap 用于请求将设备内存映射到进程地址空间。如果设备没有实现这个方法，那么 mmap 系统调用将返回-ENODEV。

```
int (*open) (struct inode *, struct file *);
```

尽管这始终是对设备文件执行的第一个操作，然而却并不要求驱动程序一定要声明这个方法。如果这个入口函数被置为 NULL,那么设备的打开操作将永远成功，但系统不会通知驱动程序。

```
int (*flush) (struct file *);
```

对 flush 操作的调用发生在进程关闭设备文件描述符副本的时候，它应该执行（并等待）设备上尚未完成的操作。请不要将它同用户程序使用的 fsync 操作相混淆。目前，flush 仅仅用于网络文件系统(Network File System, NFS)代码中。如果 flush 被置为 NULL,则它只是简单地不被调用。

```
int (*release) (struct inode *, struct file *);
```

当 file 结构被释放时，将调用这个操作。与 open 相仿，也可以没有 release 方法。

```
int (*fsync) (struct inode *, struct dentry*, int);
```

该方法是 fsync 系统调用的后端实现，用户调用它来刷新待处理的数据。如果驱动程序没有实现这一方法，fsync 系统调用就返回-EINVAL。

```
int (*fasync) (int, struct file *, int);
```

这个操作用来通知设备，它的 FASYNC 标志发生了变化。如果设备不支持异步通知，那么该入口函数可以是 NULL。

```
int (*lock) (struct file *, int, struct file_lock *);
```

lock 方法用于实现文件锁定，锁定是常规文件不可缺少的特性，但设备驱动程序几乎从来不会实现这个方法。

ssize_t (*readv) (struct file *, const struct iovec *, unsigned long, loff_t *);

ssize_t (*writev) (struct file *, const struct iovec *, unsigned long, loff_t *);

这两个方法是用来实现“分散/聚集”型的读写操作。应用程序有时需要进行涉及多个内存区域的单次读写操作，利用上面这些系统调用可完成这类工作，而不必强加额外的数据拷贝操作。

struct module *owner;

这个字段并不像 file_operations 结构中的其它部分那样，它不是一个方法。相反，它是指向“拥有“该结构的模块指针，内核使用该指针维护模块的使用计数。通常，我们并不需要实现所有的方法，只需实现最常用的方法就可以了，比如 read,write,open 等 。并且可以采用标记法来声明 file_operations 结构，如下所示：

```
struct file_operations test_fops = {
    llseek:    test_llseek,
    read:      test_read,
    write:     test_write,
    ioctl:     test_ioctl,
    open:      test_open,
    release:   test_release,
};
```

上面这个声明采用了标记化的结构初始化语法，它可以使驱动程序在结构的定义发生变化时更具可移植性，并且使得代码更加紧凑易读。标记化的初始化方法允许对结构成员进行重新排列。在某些场合下，将频繁被访问的成员放在相同的硬件缓存行上，将大提高性能。

- **struct file**

在<linux/fs.h>中定义的 struct file 是设备驱动程序中所使用的第二个最重

要的数据结构。注意，file 结构与用户空间程序中的 FILE 没有任何关联。FILE 在 C 库中定义不会出现在内核代码中；而 struct file 是一个内核数据结构，它不会出现在用户程序中。

file 结构代表一个打开的文件（它并不仅仅限于设备驱动程序，系统中每个打开的文件在内核空间都有一个对应的 file 结构)。它由内核在 open 时创建，并传递给在该文件上进行操作的所有函数，直到最后的 close 函数。在文件的所有实例都被关闭之后，内核会释放这个数据结构。注意，一个打开的文件和一个磁盘上的文件不同，后者是由 struct inode 表示的，在下文会介绍该数据结构。

struct file 中最重要的字段如下所示：

mode_t f_mode;

文件模式通过 FMODE_READ 和 FMODE_WRITE 位来标识文件是否可读或可写（或可读写)。你可能会认为要在自己的 ioctl 函数中查看这个字段来检查读/写权限，但由于内核在调用驱动程序的 read 和 write 前已经检查了权限,所以不必为这两个方法检查权限。例如，一个未得到允许的定操作在驱动程序还不知道的情况下就已经被内核拒绝了。

loff_t f_pos;

当前的读/写位置。loff_t 是一个 64 位的数，如果驱动程序需要知道文件中的当前位置，就可以读取这个值，但不能修改它(read 和 write 会使用它们接收到的最后那个指针参数来更新这一位置，而不是直接对 flip->f_pos 操作)。

unsigned int f_flags;

文件标志，如 O_RDONLY，O_NONBLOCK 和 O_SYNC。驱动程序为了支持非阻塞型操作需要检查这个标志，而其它标志很少用到。注意，检查读/写权限应该查看 f_mode 而不是 f_flags。

struct file_operations *f_op;

与文件相关的操作。内核在执行 open 操作时对这个指针赋值，以后需要处理这些操作时就读取这个指针。flip->f_op 中的值决不会为方便引用而保存起来，也就是说，你可以在任何需要的时候修改文件的关联操作，在返回给调用者之后，新的操作方法就会立即生效。例如,对应于设备号 1(/dev/null, /dev/zero 等等)的 open

代码根据要打开的次设备号替换 filp->f_op 中的操作。这种技巧允许相同的主设备号下的设备实现多操作行为，而不会增加系统调用的负担。这种替换文件操作的能力在面向对象编程技术中称为“方法重载”。

void *private_data;

open 系统调用驱动程序的 open 方法前将这个指针置为 NULL。驱动程序可以将这个字段用于任何目的或者忽略这个字段。驱动程序可以用这个字段指向已分配的数据，但是一定要在内核销毁 file 结构之前在 release 方法中释放内存，否则会造成内存泄漏。private_data 是跨系统调用时保存状态信息的非常有用的资源。

- **struct inode**

file_operations 中的大多数操作都将 inode 作为第一个参数。Linux 的 VFS 是对物理文件系统和物理设备的一个封装。inode 结构就是 VFS 与下层模块对话的重要结构。文件系统由子目录和文件构成。每个子目录或文件只能由唯一的 inode 描述。每个设备也是用 inode 来描述的。inode 是 Linux 管理文件系统的最基本单位，也就是文件系统连接任何子目录、任何文件和设备的桥梁。

struct inode {

kdev_t i_dev;

文件所有设备的设备号，如第一个 IDE 硬盘为 0x0301

unsigned long i_ino;

外存 inode 的节点号，(i_dev, i_ino)在 VFS 口是唯一的

umode_t i_mode;

表示文件类型以及存取权限

nlink_t i_nlink

连接到该文件的 link 数

uid_t i_uid;

用户标识号

gid_t i_gid;

用户组标识号

```
kdev_t        i_rdev;
```

根设备的设备号

```
off_t            i_size;
```

文件长度

```
time_t        i_atime;
```

文件访问时间

```
time_t        i_mtime;
```

文件修改时间

```
time_t        i_ctime;
```

文件创建时间

```
unsigned long    i_blksize;
```

以字节为单位的块大小，一般为 1024 字节

```
unsigned long    i_version;
unsigned  long      i_nrpages
```

文件在内存中所占页数

```
struct semaphore  i_sem;
```

信号量

```
struct inode_operations *i_op
```

指向一组针对该文件的操作函数

```
struct super_block      *i_sb;
```

指向内存中的 VFS 的超级块

```
struct wiat_queue       *i_wait;
```

在该文件上的等待队列

```
struct file_lock        *i_flock;
```

文件锁

```
struct vm_area_strct    *i_mmap;
struct page             *i_pages;
```

由文件占用页面构成的单向链，通过它可以访问内存口的文件数据

```
struct dquot            *i_dquot[MAXQUOTAS];
struct inode            *i_next, *i_prev;
```

inode 资源管理口使用的链表指针

```
struct inode            *i_hash_next, *i_hash_prev;
```

inode 缓冲区的链表指针

```
struct    inode         *i_bound_to, *i_bound_by;
struct inode            *i_mount
```

指向挂载文件系统的 inode 根目录

```
unsigned  long          i_count
```

引用记数，0 表示是空闲 inode

```
unsigned short          i_flags;
unsigned  short         i_writecount;
unsigned  char          i_lock;
```

对 inode 的加锁标志

```
unsigned  char          i_dirt;
unsigned  char          i_pipe;
unsigned  char          i_sock;
unsigned  char          i_seek;
unsigned char           i_update;
unsigned  char          i_condemned;
```

以下是一个联合结构体，是各类文件系统 inode 的特定信息

```
union  {
       struct ext2_inode_info  ext2_i;
   }u;
};
```

对以上三个数据结构了解清楚了，对驱动程序的核心数据结构也就掌握了。

8) Linux 设备驱动程序的入口与出口

与通常的 C 程序类似,驱动程序也必须一个程序的入口点(与 C 程序中的 main()相对应）和一个程序的出口点（与 C 程序中的 return 相对应），分别为如下两个。

```
module_init(test_init);          //初始化函数，即入口点
module_exit(test_cleanup);       //清除函数，即出口点
```

这两个函数通常放在源文件的末尾，并要包含头文件<linux/init.h>，其中的斜体部分为我们定义的初始化和清除函数，类似定义如下：

```
static int __init test_init(void)
{
    printk(“Module_init\n”);
    return 0;
}

static void __exit test_cleanup(void)
{
    printk(“Module removed\n”);
}
```

上述的初始化函数和清除函数什么实质工作都不做，只是输出一行提示。与 X86 体系下的驱动程序有小小的差异，我们使用了__init 和__exit 属性。__init 属性会在初始化工作完成后，丢弃初始化函数并且回收它所占用的内存。然而它只对直接静态编译到内核中的驱动程序有效，对模块化的驱动程序没有用处，因此在 ARM7 的 uClinux 中，我们通常加上这一属性。__exit 属性用在直接静态编译到内核中的驱动程序时，它将忽略所标记的函数，因为它不可能被卸载,而用在模块中时则没有任何作用。

3、Linux 字符设备驱动程序基本框架

在嵌入式系统中，我们碰到的大多数驱动程序都是以字符设备为基础的，而且字符设备驱动程序入门简单，因此，本说明书中着重讲解字符设备驱动程序。在本小节中，我们可以看到一个什么都不做的驱动程序，它只在操作系统调用相应功能的时候，输出一行提示信息，证明它正在工作。通过这个简单的字符设备驱动程序，我们可以更清晰地看到一个驱动程序的基本框架。

- 驱动程序的预定义和头文件包含部分

```
#ifndef __KERNEL__    /* 定义__KERNEL__符号，以指明这是内核空间的程序*/
#define __KERNEL__
#endif
/* 而且在 uClinux 中，驱动程序是静态编译至内核的，并不是以模块方式动态加载的，因此不需要定义 MODULE 符号*/
#include <linux/module.h>
#include <linux/kernel.h>   /* For printk() */
#include <linux/config.h>
#include <linux/fs.h>       /* For struct file_operations */
#include <linux/types.h>
#include <linux/sched.h>
#include <linux/init.h>     /* For module_init & module_cleanup */

#define TEST_MAJOR  253
```

在上面的代码片段中，我们可以看到许多驱动程序都需要的一些预处理符号定义，如__KERNEL__这个预处理符号在编译模块化内核代码时都必须定义。然后还要包含一些内核常用的头文件，代码注释中已经给出了部分头文件包含的理由。最后一行定义了该驱动程序的主设备号（MAJOR）253。

- 函数声明和 file_operations 结构的填写

```
static ssize_t test_read(struct file *filp, char* buf, size_t count, loff_t *f_pos);
static ssize_t test_write(struct file *filp, const char* buf, size_t count, loff_t *f_pos);
static loff_t test_llseek(struct file *filp, loff_t off, int whence);
static int test_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg);
static int test_open(struct inode *inode, struct file *filp);
static int test_release(struct inode *inode, struct file *filp);

/* 以下为驱动程序口的核心数据结构，它告诉系统内核，该设备的几个操作所对应的函数入口点在该结构体的定义中采用了标记化的结构初始化语法 */
struct file_operations test_fops = {
    llseek:    test_llseek,
    read:  test_read,
    write: test_write,
    ioctl: test_ioctl,
    open:  test_open,
    release: test_release,
```

```
};
```

- 具体操作函数的实现

本驱动程序的目的是描述字符设备驱动程序的基本框架，因此，所有操作函数都不做任何具体的工作，只是打印出一句提示信息。需要注意的是，在驱动程序中，打印输出语句并不使用我们常用的 printf()，而是使用 printk()函数。这是因为在内核模块中不能使用用户空间定义的库函数。本驱动程序实现了 6 个操作函数，test_read()，test_write()，test_ioctl()，test_open()，test_release()，test_llseek()。源程序如下：

```
/* 驱动程序 open 操作对应的操作函数 */
static int test_open(struct inode *inode, struct file *filp)
{
    printk("Device TEST open!\n\r");
    return 0;
}

/* 驱动程序 llseek 操作对应的操作函数  */
static loff_t test_llseek(struct file *filp, loff_t off, int whence)
{
    printk("Device TEST seek!\n\r");
    return 1;
}

/* 驱动程序 write 操作对应的操作函数   */
static ssize_t test_write(struct file *flip, const char *buf, size_t count,
loff_t *f_pos)
{
    printk("Device TEST write!\n\r");
    return count;
}

/* 驱动程序 ioctl 操作对应的操作函数   */
static int test_ioctl(struct inode *inode, struct file *filp, unsigned int
cmd, unsigned long arg)
{
    printk("Device TEST ioctl!\n\r");
    return 1;
}
```

```
/* 驱动程序 release 操作对应的操作函数 */
static int test_release(struct inode *inode, struct file *flip)
{
    printk("Device TEST release!\n\r");
    return 0;
}

/* 驱动程序 read 操作对应的操作函数 */
static ssize_t test_read(struct file *filp, char *buf, size_t count, loff_t
*f_pos)
{
    printk("Device TEST read!\n\r");
    return count;
}
```

- 驱动程序的初始化函数

```
/* 驱动程序真正的初始化过程，它由本程序最后的 module_init 指明由于它使用
了__init 关键字，而且该驱动程序是以静态方式编译到内核中的，因此当驱动程序
初始化完成后，它所占据的内存空间会被释放  */
static int __init test_init_module(void)
{
    int result;

    printk("Device TEST init_module\n\r");
    result = register_chrdev(TEST_MAJOR, "TEST", &test_fops);

    if (result < 0) {
        printk("TEST can't get major number\n\r");
        return result;
    }
    return 0;

}

/* 驱动程序退出过程，它由本程序最后的 module_cleanup 指明由于它使用了
__exit 关键字，并且，它也是静态编译至内核中的，因此，下面这段代码都不会被
执行，因为以静态方式编译到内核的驱动程序是无法卸载的 */
static void __exit test_cleanup_module(void)
{
    unregister_chrdev(TEST_MAJOR, "TEST");
    printk("Device TEST remove\n\r");
}
```

```
module_init(test_init_module);          //驱动程序入口函数
module_exit(test_cleanup_module);       //驱动程序出口函数

EXPORT_NO_SYMBOLS;              /*为了不把该驱动程序中定义的符号，变量名等引
                              入内核空间，我们需要在驱动程序中加入这个宏*/
MODULE_LICENSE("GPL");                  //驱动程序的版权说明
MODULE_AUTHOR("51EDA");                 //指明该驱动程序模块的作者
```

至此，我们的第一个驱动程序已经完成，完整的源代码可以在附录中得到，下一小节，我们将讨论驱动程序的编译和内核设置工作。

4、字符设备驱动程序的编译与测试

1) 编译驱动程序

- 修改 Makefile 文件

在uClinux中,字符设备驱动程序通常放在Clinux-dist/linux-2.4.x/driver/char目录下。所以我们也把上一小节所编的程序放到这个目录下，假设文件名为test_dev.c。为了把这个驱动程序编译到内核中去，我们需要修改uClinux-dist/linux-2.4.x/driver/char 目录下的 Makefile 文件。在该文件的 210 行左右，可以看到如下代码

```
#
#       uClinux    drivers
#
obj-$(CONFIG_SERIAL_AMBA)       += serial_core.o
obj-$(CONFIG_68328_SERIAL)      += 68328serial.o
obj-$(CONFIG_SERIAL_DSC21)      += serial_dsc21.o
obj-$(CONFIG_SERIAL_ATMEL)      += serial_atmel.o
```

在以上代码中，“#”代表是注释，而

```
obj-$(CONFIG_SERIAL_AMBA)    += serial_core.o
```

则表示当 CONFIG_SERIAL_AMBA 选项被选中时，serial_core.o 模块将被编译进系统内核中。因此，为了把我们的驱动模块编译进内核中，我们需要加上一行代码：

```
obj-$(CONFIG_TEST_DEV)       += test_dev.o
```

这行代码表示，当 CONFIG_TEST_Driver 被选中时候，test_dev.o 模块将被编译

至内核中。

- 修改 Config.in 文件

在上一小节中，我们提到了当 CONFIG_TEST_Driver 被选中时，我们自己编写的驱动模块才能编译到内核中,那 CONFIG_TEST_Driver 又需要在什么地方设置呢？我们通过修改uClinux-dist/linux-2.4.x/driver/char 目录下的Config.in 文件来添加 CONFIG_TEST_Driver 选项。用 vi 或其它编辑软件打开 Config.in 文件后，可以看到如下代码

```
#######################################################3
#      uClinux    options
#
if [“$CONFIG_EXCALIBUR” = “y”]; then
   bool    ‘Nios serial support’  CONFIG_NIOS_SERIAL
   bool    ‘Nios SPI  Device  support’    CONFIG_NIOS_SPI
fi
```

我们在“#”的下一行添加一行代码

```
   bool    ‘44B0X Test Device Driver’  CONFIG_TEST_DEV
```

这样，我们就可以在 make menuconfig 的配置菜单中选择将 test_dev 驱动编译进内核。

- 配置内核

将工作目录切换到 uClinux-dist 目录下，执行内核配置程序 make menuconfig。按上文的编译步骤进入到如下图所示的界面，并选择红框中的 Character devices 选项：

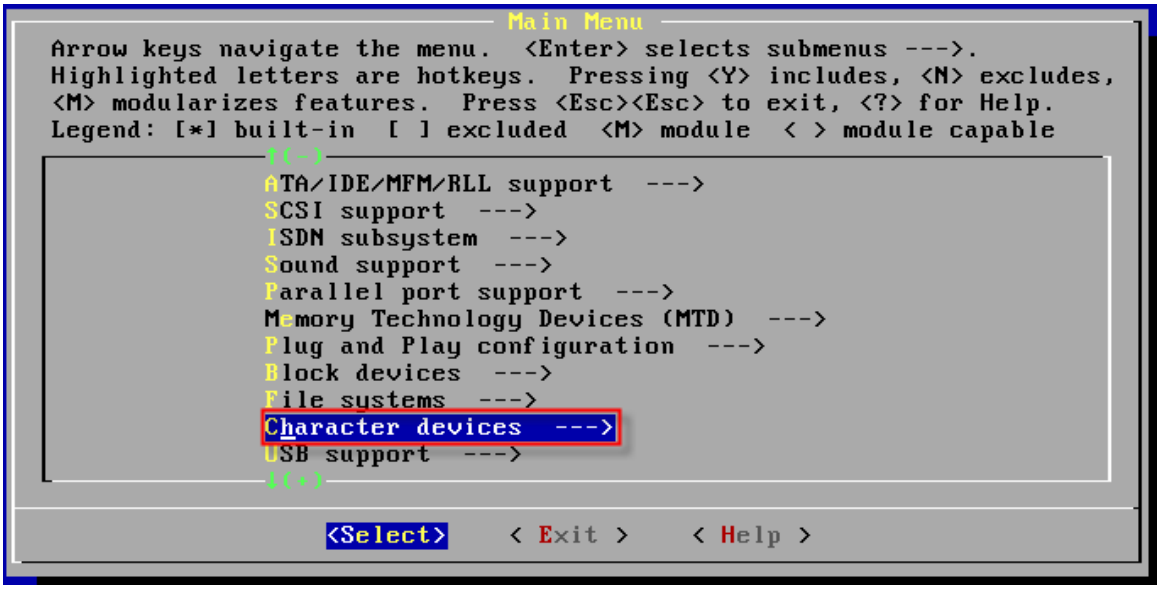


图 16-5

这时可以看到我们刚才添加的选项，如下图红线处所示，用空格选中该选项：

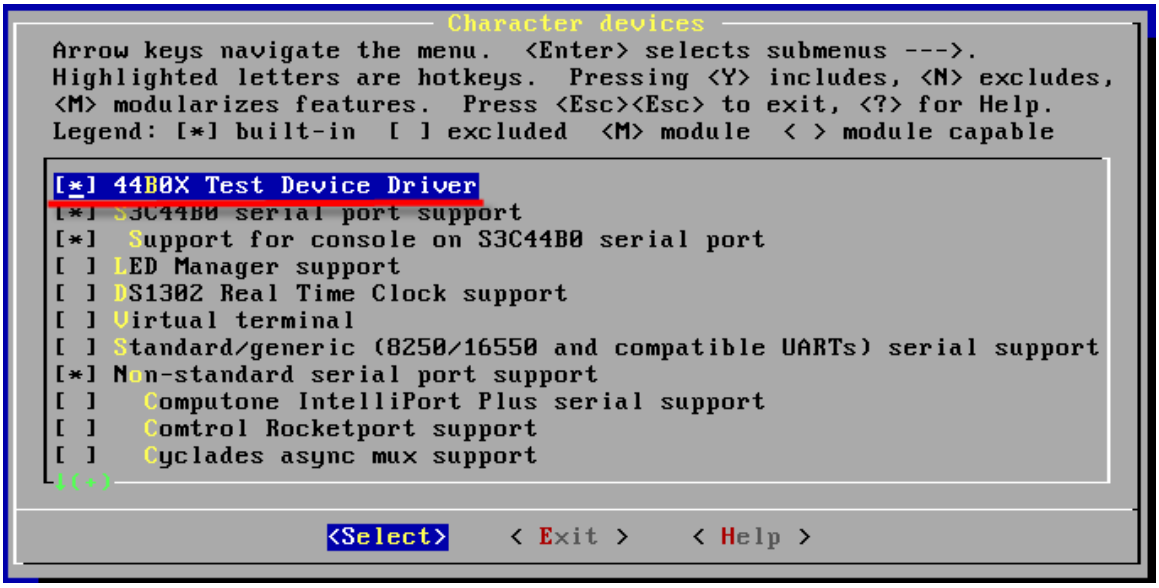


图 16-6

两次选择“Exit”后，选择保存设置后退出。

- 添加设备节点

在前文中已经提过，uClinux 采用的是只读的 romfs 文件系统，也就是说当 uClinux 被下载到开发板上后，我们就不能动态地建立和删除设备节点，因此我们需要修改 uClinux-dist/vendor/Samsung/44B0X/Makefile 文件，使得在编译系统内核时，自动生成相应的设备节点。在该 Makefile 文件中，我们可以看到如下代码行：

```
DEVICES = \
    tty,c,5,0  console,c,5,1 cua0,c,5,64    cua1,c,5,65 \
    \
    mem,c,1,1  kmem,c,1,2     null,c,1,3     test,c,253,0\
    zero,c,1,5 random,c,1,8   urandom,c,1,9   rtc,c,10,135\
    \
    ram0,b,1,0 ram1,b,1,1 \
```

在上述代码行中，带下划线的部分是我们需要添加的，它表示在/dev 目录下建立名为 test 的设备节点，它的主设备号(MAJOR)为 253，而次设备号(MINOR)为 0。

- 编译

将当前工作目录设为 uClinux-dist 后，按正常的编译顺序编译内核，具体步骤请查看第十一章的内容。当编译通过，没有错误后，我们的驱动程序 test_dev.o 就成功地加进内核中了，下一小节将说明如何测试或使用该驱动程序。

2) 测试程序

当把上面的 uClinux 映象文件下载到开发板，启动板子，我们可以看到下图的提示信息，红线处标明了 Teset Device 驱动加载到内核到的提示信息。并且在命令提示符下输入 ls /dev -l 命令,当看到有 test 这设备节点时，我们就可以使用该驱动程序了。我们在用户空间下编写一个小的测试程序。

- 编写测试程序

在 uClinux-dist/user 目录下建立一个目录 driver，并在该目录下编辑测试程序。假设程序名为 test_driver.c，源程序分析如下：

```
#include <stdio.h>      /* printf()函数 */
#include <string.h>     /*  字符串操作  */
#include <fcntl.h>      /* For O_WRONLY O_RDONLY O_RDWR */
#include <sys/ioctl.h>  /* For ioctl() */

#define  IOCTL_TEST 1   /* 为了测试 ioctl()驱动功能，我们定义了一个 ioctl 的操作命令*/

/* 驱动程序的写功能测试 */
void write_test()
{
```

```
    int fd;
    char *test_char = "Write Test";      /* 测试字符串 */
    fd = open("/dev/test", O_WRONLY);    /* 以只写方式打开/dev/test 设备 */
    if (fd < 0) {                        /* 如果设备打开失败，输出提示信息并退出 */
        printf("Device 'TEST' can't be opened\n");
        return;
    }

    /*  对设备进行写操作，写入"Write Test"字符串 */
    write(fd, test_char, strlen(test_char));
    close(fd);                /* 关闭设备 */
    return;
}

/* 驱动程序的读功能测试 */
void read_test()
{
    int fd;
    char buf[2];

    fd = open("/dev/test", O_RDONLY);    /* 以只读方式打开/dev/test 设备 */
    if (fd < 0) {                        /* 如果设备打开失败，输出提示信息并退出 */
        printf("Device 'TEST' can't be opened\n");
        return;
    }
    read(fd, buf, sizeof(buf));      /* 对设备进行读操作 */

    close(fd);
    return;
}

/* 驱动程序的 ioctl 功能测试 */
void ioctl_test()
{
    int fd;
    fd = open("/dev/test", O_RDWR); /* 以读写方式打开/dev/test 设备 */
    if (fd < 0) {            /* 如果设备打开失败，输出提示信息并退出 */
        printf("Device 'TEST' can't be opened\n");
        return;
    }
    ioctl(fd, IOCTL_TEST);       /* 对设备进行 ioctl 操作,操作命令为 IOCTL_TEST */
    return ;
}

/* 驱动程序的 lseek 功能测试 */
void lseek_test()
{
```

```
    int fd;
    fd = open("/dev/test", O_RDONLY);   /* 以只读方式打开/dev/test 设备 */
    if (fd < 0) {                /* 如果设备打开失败，输出提示信息并退出 */
        printf("Device 'TEST' can't be opened\n");
        return;
    }

/* 对设备进行 lseek 操作， 偏移量为 10，起始位置为文件首部*/
    lseek(fd, 10, SEEK_SET);
    return ;
}


/*  主函数      */
int main(int argc, char* *argv)
{
    /*  命令行输入格式判断，如果不符合要求则给出提示信息 */
    if (argc == 1) {
        printf("Usage test_driver [read|write|ioctl|lseek]\n");
        return 0;
    }
    /*  命令参数为 write 时，执行写操作测试 */
    if (!strcmp(argv[1], "write"))
        write_test();
    /*  命令参数为 read 时，执行读操作测试 */
    else if (!strcmp(argv[1], "read"))
        read_test();
    /*  命令参数为 ioctl 时，执行 IO 控制操作测试 */
    else if (!strcmp(argv[1], "ioctl"))
        ioctl_test();
    /*  命令参数为 lseek 时，执行 lseek 操作测试 */
    else if (!strcmp(argv[1], "lseek"))
        lseek_test();
    else
        printf("Usage test_driver [read|write|ioctl|lseek]\n");

    return 0;
}
```

- 编辑 Makefile 文件

需要编写或修改两个 Makefile 文件

为了编译这个测试程序，首先要在 uClinux-dist/user/driver 目录下编写一个 Makefile 文件，如下所示：

```
EXEC = test_driver
OBJS = test_driver.o
```

```
all: $(EXEC)

$(EXEC): $(OBJS)
   $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

romfs:
   $(ROMFSINST) /bin/$(EXEC)

clean:
   -rm -f $(EXEC) *.elf *.gdb *.o
```

为了在 uClinux 中添加我们刚编写的测试程序，我们需要编辑 uClinux-dist/user/Makefile 文件，在 dir_y += hello 的下一行加上一行代码

```
dir_y += driver
```

它表示，在编译时，把 driver 目录加到编译目录列表中。至此，文件的编辑和修改工作已经完成，下面是编译过程，按前文的步骤，重新编译 uClinux，并把生成的映象文件下载到开发板中，就可以进行下一步测试工作了。

- 测试

在命令行提示符下输入 test_driver read，进行读操作测试，测试结果如下图所示：

```
Sash command shell (version 1.1.1)
/> test_driver read
Device TEST open!
Device TEST read!
Device TEST release!
/>
```

图 16-7

对照 test_dev.c 驱动程序，我们从上图中可以清晰地看到，读操作调用了驱动程序中的三个基本操作，分别是 open、read 和 release。同理，我们进行 write、ioctl 和 lseek 的操作测试，测试结果分别如下：

```
/> test_driver write
Device TEST open!
Device TEST write!
Device TEST release!
/>
```

图 16-8

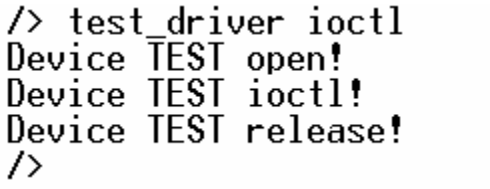


图 16-9

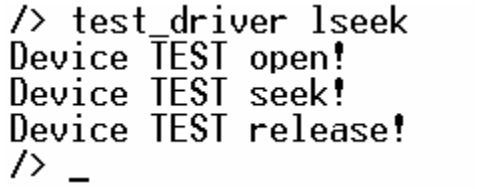


图 16-10

Linux 字符设备驱动程序 scull

在本小节中的 scull 驱动程序是一个简单的而完整的驱动程序，它的功能是按用户的需要，在 RAM 中开辟一个空间来存储用户输入的内容，如果用户不重启开发板，不重新往里写入数据或是执行 RESET 操作的话，用户输入的内容可以一直保存在存储空间中，我们可以随时将其中的内容读出来。

- 源程序

  scull 驱动程序和注释如下：

```
#ifndef __KERNEL__
#define __KERNEL__
#endif

#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/ioctl.h>
#include <linux/fs.h>
#include <linux/types.h>
#include <linux/slab.h>     /* For  kmalloc() */
#include <asm/uaccess.h>    /* For copy_to_user */

#define SCULL_MAJOR 254     /* SCULL 设备的主设备号 */
#define SCULL_MAGIC SCULL_MAJOR
/*  以下定义了 SCULL 设备 ioctl 的几种操作，
    分别为 SCULL 设备的 RESET，即清除缓冲区的内容
    返回 SCULL 设备中数据的长度          */
```

```
#define SCULL_RESET _IO(SCULL_MAGIC, 0)
#define SCULL_QUERY_MSG_LENGTH  _IO(SCULL_MAGIC, 1)

/*  定义 SCULL 设备的数据结构 */
struct Scull_Dev {
    char *data;             /* 数据的存储区域 */
    unsigned long size;     /* SCULL 设备最大可以存储的字节数
                               在 scull_open 操作中进行初始化 */
    unsigned int usage;     /* 当 SCULL 设备被占用时，该位置 1 */
} scull;

/* 以下为函数原型 */
static int scull_open(struct inode *inode, struct file *filp);
static int scull_release(struct inode *inode, struct file *filp);
static ssize_t scull_write(struct file * filp, const char *buffer, size_t count, loff_t
*f_pos);
static ssize_t scull_read(struct file *filp, char *buffer, size_t count, loff_t
*f_pos);
static int scull_ioctl(struct inode *inode, struct file *filp, unsigned int cmd,
unsigned long arg);

/* 填写驱动程序中最重要的 file_operations 结构，指明各种处理的函数入口 */
struct file_operations scull_fops = {
    read:   scull_read,
    write:  scull_write,
    open:   scull_open,
    ioctl:  scull_ioctl,
    release:scull_release,
};

/* open 操作的入口函数，初始化 scull.size  */
static int scull_open(struct inode *inode, struct file *filp)
{
    scull.size = 0x10000;   /* 大小为 64KB */
    return 0;
}

/* release 操作的入口函数，不做任何事情 */
static int scull_release(struct inode *inode, struct file *filp)
{
    return 0;
}

/* read 操作的入口函数 */
static ssize_t scull_read(struct file *filp, char *buffer, size_t count, loff_t *f_pos)
{
    int length;
```

```
    /* 当要读的字节数<0 时，返回无效操作错误 */
    if (count < 0)
        return -EINVAL;
    /* 当 SCULL 设备被占用时，返回设备忙错误 */
    if (scull.usage)
        return -EBUSY;
    /* 当 SCULL 设备为空时，返回 0，表示没有读到任何数据 */
    if (scull.data == NULL)
        return 0;

    /* 将 SCULL 设备置为占用，当另一个操作同时要使用该设备时，
        就会返回设备忙错误  */
        scull.usage = 1;
        /* 计算设备中数据的长度 */
    length = strlen(scull.data);
    /* 如果要读取的数据比设备中的实际数据还要多，则只读取设备中的全部数据 */
    if (length < count)
        count = length;
        printk("count = %d\n", count);

      /*  copy_to_user()表示把数据从内核空间拷贝到用户空间中，其中 buffer 是用户空间
提供的缓冲区指针，scull.data 是内核空间的缓冲区指针，从而实现内核和用户空间数据的传
输，更多关于 copy_to_user 函数的内容请参看相应的文档 */
    if (copy_to_user(buffer, scull.data, count)) {
            printk("copy_to_user error\n\r");
            return 0;
    }
    /* 将 SCULL 设备被占用标志清 0，其它应用程序现在可以使用该设备了 */
    scull.usage = 0;
    /* 偏移位置加上这次读操作的字节数 */
        *f_pos += count;
    return count;
}

/* write 操作的入口函数 */
static ssize_t scull_write(struct file *filp, const char *buffer, size_t count, loff_t
*f_pos)
{
    /* 当要写的字节数<0 时，返回无效操作错误 */
    if (count < 0)
        return -EINVAL;
    /* 如果当前的写位置+这次操作需要写的字节数超过了 SCULL 设备的最大空间 100KB,
       则输出提示信息设备满，并返回无效操作错误 */
    if (*f_pos + count > scull.size) {
        printk("Device is full\n\r");
        return -EINVAL;
    }
```

```
    /* 当 SCULL 设备被占用时，返回设备忙错误 */
    if (scull.usage) {
        printk("scull Device is busy!\n\r");
        return -EBUSY;
    }
/* 将 SCULL 设备占用标志置 1，当另一个操作同时要使用该设备时，就会返回设备忙错误  */
    scull.usage = 1;
```

/* 如果 scull.data 没有分配过存储空间，则分配存储空间，在内核程序中，不能使用 malloc()函数来分配空间，只能使用 kmalloc()函数，它有两个参数，第一个参数为存储区的大小，第二个参数是 GFP_KERNEL 表示优先级，通常用 GFP_KERNEL，其它优先级和作用如下所示：

GFP_BUFFER

它的优先级最低，如果所申请的内存不足，或已经被占用，它并不释放那些不可用的内存页面。

GFP_ATOMIC

尽可能马上分配存储空间，在这个优先权的情况下，即使所申请的内存页面不可用，该操作也不会睡眠，通常在中断服务程序中需要这样的特性。在系统中，会保留有一定量的内存页面用于这些存储空间的申请。

GFP_KERNEL

在内核程序中正常申请存储空间的优先级，它不会占用为 GFP_ATOMIC 优先级保留的内存页面，使用 GFP_KERNEL 优先权允许 kmalloc 函数在系统空闲内存低于水平线 min_free_pages 时延迟分配函数的返回。当空闲内存太少时，kmalloc 函数会使当前进程进入睡眠，等待空闲页的出现。这是最常用的优先级。

GFP_USER

目前，与 GFP_KERNEL 同样的效果

GFP_NFS

优先权 GFP_NFS 与 GFP_KERNEL 类似，但它会使得 NFS 文件系统缩减空闲列表到 min_free_pages 值以下。显然，为使驱动程序“更快”而用 GFP_NFS 优先权取代 GFP_KERNEL 优先权会降低整个系统的性能。

GFP_DMA

GFP_DMA 标志位要和 GFP_KERNEL 和 GFP_ATOMIC 优先权一起使用来分配用于直接内存访问(DMA)的内存页

```
        */
        if (!scull.data) {
            scull.data = kmalloc(sizeof(char)*(scull.size), GFP_KERNEL);
            if (!scull.data) {
                printk("scull.data kmalloc error\n\r");
            return -ENOMEM;
        }
    }
```

/* copy_from_user()表示把数据从用户空间拷贝到内核空间中，其中 buffer 是用户空

```
间提供的缓冲区指针，scull.data 是内核空间的缓冲区指针，*f_pos 是当前写入位置的偏移量。
利用这个函数实现用户空间到内核空间数据的传输。*/
    if(copy_from_user(scull.data + *f_pos, buffer, count + 1)) {
        printk("copy_from_user error\n\r");
        return -ENOMEM;
    }
    /* 将 SCULL 设备被占用标志清 0，其它应用程序现在可以使用该设备了 */
    scull.usage = 0;
    /* 维护当前写入位置的偏移量，即加上本次操作写入的字节数  */
    *f_pos += count;
    return count;
}

/* write 操作的入口函数，在这个函数中，我们只实现了两个功能
   1  清空设备中的所有数据 SCULL_RESET
   2  返回设备中数据的长度 SCULL_QUERY_MSG_LENGTH
*/
static scull_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned
long arg)
{
    int ret = 0;
    switch(cmd) {
    /* 当操作命令为 SCULL_RESET 时，清空设备中的数据 */
    case        SCULL_RESET:
/* 将 SCULL 设备占用标志置 1，当另一个操作同时要使用该设备时，就会返回设备忙错误  */
                scull.usage = 1;
        /* 用 kmalloc 分配的内存要用 kfree()函数来释放 */
                kfree(scull.data);
                scull.data = NULL;
        /* 将 SCULL 设备被占用标志清 0，其它应用程序现在可以使用该设备了 */
                scull.usage = 0;
                        printk("Buffer has been released\n\r");
                break;

/* 当操作命令为 SCULL_QUERY_MSG_LENGTH 时，返回设备中数据的字节数 */
    case        SCULL_QUERY_MSG_LENGTH:
                /* 当设备中没有数据时，返回 0 */
                        if (scull.data == NULL)
                    return 0;
                else
                    return strlen(scull.data);
                break;

    /* 如果操作命令没有被定义，则返回没有实现错误 */
        default:
                return -ENOTTY;
        }
```

```
    return ret;
}

/* SCULL 设备的初始化函数 */
static int __init scull_init_module(void)
{
    int result;

    /* 注册字符设备"scull"，它的主设备号 SCULL_MAJOR 为 254 */
        result = register_chrdev(SCULL_MAJOR, "scull", &scull_fops);
    if (result < 0) {
        printk("Device Scull can't get major\n\r");
        return result;
        }
    return 0;
}

static void __exit scull_cleanup_module(void)
{
    unregister_chrdev(SCULL_MAJOR, "scull");
}

module_init(scull_init_module);     //驱动程序入口函数
module_exit(scull_cleanup_module);  //驱动程序出口函数

EXPORT_NO_SYMBOLS;          //为了不把该驱动程序中定义的符号，变量名等
                            //引入内核空间，我们需要在驱动程序中加入这个宏
MODULE_AUTHOR("51EDA");         //驱动程序的作者
MODULE_LICENSE("GPL");          //驱动程序的版权说明
```

- 创建设备节点

在驱动程序中，我们已经知道 SCULL 设备的主设备号是 254，而且它是字符型设备，因此按照前文的步骤，修改 uClinus-dist/vendor/Samsung/44B0X/Makefile 文件，在相应的位置加上设备节点信息 scull,c,254,0。

- 驱动程序的测试

我们同样编写了一个简单的设备驱动测试程序，该程序的功能如下：

1、scull_test write，参数 write 表明是往设备里写入数据，空行表示输入结束。

2、scull_test read,，参数 read 表明是从设备里读数据。

3、scull_test reset，参数 reset 用于清除设备里的全部数据。

源程序如下：

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <linux/types.h>
#include <sys/ioctl.h>

#define SCULL_MAJOR 254 /* SCULL 设备的主设备号 */
#define SCULL_MAGIC SCULL_MAJOR
/*  以下定义了 SCULL 设备 ioctl 的几种操作，分别为 SCULL 设备的 RESET，即清除缓冲区的
内容返回 SCULL 设备中数据的长度          */
#define SCULL_RESET _IO(SCULL_MAGIC, 0)
#define SCULL_QUERY_MSG_LENGTH  _IO(SCULL_MAGIC, 1)

/* 定义缓冲区的大小，64KB */
#define MAXLINE 65536

/* 写操作测试 */
int write_test()
{
    int fd, len;
    char buf[MAXLINE];

    fd = open("/dev/scull", O_RDWR|O_APPEND);
    if (fd < 0) {
        printf("Device cannot be opened\n");
        return 0;
    }
    while (gets(buf)) {
        if (!strlen(buf))
            break;

        if ((len = write(fd, buf, strlen(buf))) <= 0) {
            printf("Write error\n");
            return 0;
        }
    }
    close(fd);

    return 0;
}
/* RESET 操作测试 */
int reset_test()
{
    int fd;

    fd = open("/dev/scull", O_WRONLY);
    if (fd <0) {
```

```
        printf("Device scull can't be opened\n");
        return 0;
    }

    ioctl(fd, SCULL_RESET);
    return 0;
}

/* read 操作测试 */
int read_test()
{
    int fd, len;
    char buf[MAXLINE];

    fd = open("/dev/scull", O_RDONLY);
    if (fd < 0) {
        printf("Device scull cannot be opened\n");
        return 0;
    }
    len = ioctl(fd, SCULL_QUERY_MSG_LENGTH, NULL);
    if ((read(fd, buf, len)) != len) {
        printf("Read error\n\r");
    return 0;
    }
    buf[len] = '\0';
    printf("buf = %s\n", buf);
    close(fd);
    return 0;
}

int main(int argc, char* *argv)
{
    if (argc == 1) {
        printf("Usage: scull_test [read|write|reset]\n");
        return 0;
    }

    if (!strcmp(argv[1], "write"))
        write_test();
    else if (!strcmp(argv[1], "read"))
        read_test();
    else if (!strcmp(argv[1], "reset"))
        reset_test();
    else printf("Usage: scull_test [read|write|reset]\n");

    return 0;
}
```

- 测试文件的编译

在 uClinux-dist/user 目录下新建一名为 scull_test 的目录，将上述测试文件放到该目录下，并 uClinux-dist/user/scull_test 目录下新建一 Makefile 文件，文件内容如下所示：

```
EXEC = scull_test
OBJS = scull_test.o

all: $(EXEC)

$(EXEC): $(OBJS)
  $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

romfs:
  $(ROMFSINST) /bin/$(EXEC)

clean:
  -rm -f $(EXEC) *.elf *.gdb *.o
```

然后修改 uClinux-dist/user 目录下的 Makefile 文件，在 dir_y += hello 的下一行添加一行代码：

```
dir_y      +=      scull_test
```

然后重新编译内核，并将生成的映象文件下载到开发板中，就可以进行下一步测试工作了。

- 驱动程序的测试

1、写测试

在命令行提示符下输入写测试命令

```
scull_test write
```

然后键入字符，可以分成多行输入，空行表示输入结束，所有用户输入的字符都将保存在 SCULL 设备中（内存），可以在任何时候读出来并显示。

2、读测试

在命令行提示符入输入读测试命令：

```
scull_test read
```

可以把 SCULL 设备中所存的字符输出。测试过程如下图所示：

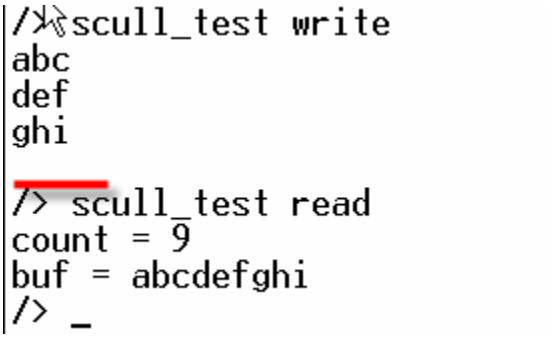


图 16-11

输入数据分成三行输入，其中划线处为空行（直接按回车），表示输入结束。在执行 scull_test read 命令后，会提示共有 9 个字符，设备中内存的数据为“abcdefghi”。

3、RESET 测试

如果需要把 SCULL 设备中的全部数据清除,则要用到 RESET 参数,执行测试命令：

scull_test reset

测试结果如下图所示

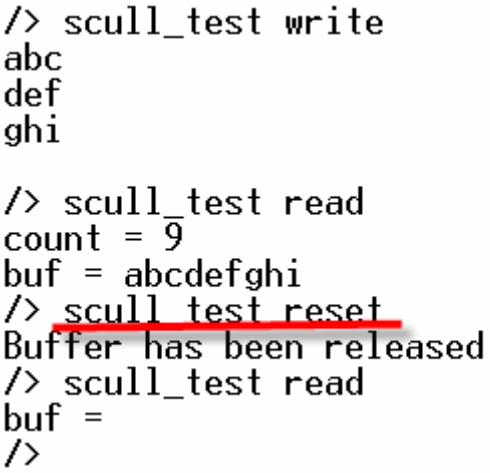


图 16-12

当执行划线处的 scull_test rest 命令后，设备中的数据被全部清除。我们再读设备的时候，输出为空。

4、cp 命令测试

由于 SCULL 是一个设备，而且我们已经编写了合适的驱动程序，因此我们可以使

用 Linux 下的一些命令来直接使用该设备。以 cp 命令为例:

```
cp      /etc/rc    /dev/scull
```

这时/etc/rc 文件就被存到 SCULL 设备中了，我们可以用 scull_test read 指令来查看，测试结果如下图所示：

```
Sash command shell (version 1.1.1)
/> cp /etc/rc /dev/scull
/> scull_test read
count = 369
buf = hostname Samsung
/bin/expand /etc/ramfs.img /dev/ram0
mount -t proc proc /proc
mount /var
mkdir /var/config
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
mkdir /var/empty
mount /mnt/yaffs
mount /mnt/jffs2
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.255.255.0 lo
ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up
portmap &
cat /etc/motd

/> _
```

图 16-13

从上图中可以看出，/etc/rc 文件确实被存储到 SCULL 设备中了，因此我们的驱动程序是可以正常工作的。

本章只要简述地说明了 Linux 设备驱动程序的基本编写方法，并以字符设备为例，给出了相应的例子，但对于想深入了解 Linux 驱动程序的用户来说，最好参阅相关的书籍。

本章中的所有程序都已经放在指定目录下

第十七章 uClinux 环境结构简单分析

1、系统目录结构

- **目标板上的 uclinux 目录结构**

我们在目标板上运行 uclinux 后，用 ls 指令，我们可以看到如下的目录结构

bin dev etc home lib mnt proc sbin tmp usr var

因为目标板上烧制的是一个linux 操作系统，所以它的文件目录结构和主机上的linux基本类似，目录的意义作用也基本类似。但目标板上文件系统和PC系统区别还是非常大的，例如板上文件系统只有几百K字节，而主机上的文件系统则有几个G之大。我们用下表来作一个对比：

目录	uClinux	Linux
bin	各种可执行文件	同左
dev	各种设备驱动	同左
etc	含启动脚本及各个应用程序的配置文件	同左
home	在豪华板上无用	多用户的工作目录
lib	系统库文件，但由于 uClinux 采用静态编译，因此无用	系统库文件
mnt	文件系统的挂载点	同左
proc	proc 文件系统	同左
sbin	是一个链接，指向/bin，也就是说，当我们进入 sbin 这个目录时，事实上进入的是 bin 目录	系统可执行文件的存放目录
tmp	同样也是一个链接，指向/var/tmp，也就是，当我们进入	临时文件目录

	/tmp 这个目录时，事实上进入的是/var/tmp 这个目录	
usr	无用	重要的系统目录，含用户安装的应用程序及 LINUX 内核源代码
var	我们板上采用的是 romfs 文件系统，是只读的，而 var 是可写的目录，采用的是 ext2 文件格式	日志文件等常改变文件的存放目录

2、如何实现启动 uclinux 后自动运行某一程序（rc 文件分析）

嵌入系统通常都要求系统启动后自行运行指定的程序，而不需要人为地干预。uclinux 启动后，会根据一个名为 rc 的文件，依次执行其中的指令，因此我们只需在这个文件中加入我们应用程序的文件名，就可以实现自动运行。豪华板上所对应的 rc 文件位于 uClinux-dist/vendors/Samsung/44B0X 目录下。我们也可以在目标板上查看这个文件。在目标板上输入指令：

cat /etc/rc

可以看到 rc 文件的内容如下：

```
hostname Samsung                          设置目标板的主机名
/bin/expand /etc/ramfs.img /dev/ram0
mount -t proc proc /proc                  以下两项为挂载文件系统
mount -t ext2 /dev/ram0 /var              /var 目录为可读写目录,因此它的文件
                                          格式为 ext2
mkdir /var/config                         在目标板的内存中建立系统所需的
                                          各目录
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
mkdir /var/empty
```

ifconfig lo 127.0.0.1　　设置目标板的网络参数

route add -net 127.0.0.0 netmask 255.255.255.0 lo

ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up 启动板上的网卡并将它的 IP 地址设置为 192.168.1.100，改变这个文件中的 IP 地址就可以改变目标板上的 IP 地址

cat /etc/motd　　显示 uclinux 的 LOGO。

如果我们需要在 uclinux 启动完后，自动运行我们的 hello 程序，则只需修改主机上 uClinux-dist/vendors/Samsung/44B0X/rc 文件，在其最后添加一行

hello

然后重新编译 uclinux 并下载到目标板上，当目标板重启完成后，就会自动运行 hello 程序，输出一行“Hello World”。

第十八章　将 uClinux 移植到其他平台

1、 移植（Port)的概念

Linux 的开源性决定了它可以在多种平台上运行,但每一种平台都有一些硬件上的差异，因此我们需要改动源代码，使其能在不同的平台上运行。

- 基于处理器的移植

这种类型的移植要求从支持处理器的编译器开始。这是最主要也是最困难的一步。通常当一个操作系统（即便是一个应用程序）要运行于一处理器，都需要特定的编译器把操作系统（应用程序）编译成处理器可识别的字节码。针对 Linux 系统而言，由于 GNU 套件支持大量的处理器。GNU 编译器 GCC 在设计时就已经考虑跨平台的问题，所以在进行 GCC 移植时我们可以不考虑前端高级语言解析的部分〔即针对 C 语言等解析的过程），而只需要考虑低端的移植（主要针对处理器部分），即告诉 GCC 在编译时怎样形成汇编代码，怎样满足处理器体系结构下的参数传递，怎样针对处理器进行流水线优化等等。

基于处理器的移植还包括操作系统的移植（假如嵌入式设备不需要操作系统，则编译器完成后就可以进入编写应用的阶段）。任何操作系统都有一定的代码是同处理器相关的，而操作系统为了增加运行效率，大都使用了大量特定处理器提供的底层功能支持。这些与特定处理器相关的部分最终都必须修改，使其适用于新的处理器。(下文的描述主要是这部分的移植)

另外从编写应用的角度来看，还必须提供函数库。因此函数库的移植也是必须的。

- 基于平台的移植

这种移植相对于处理器的移植而言所处的开发层次更高，主要在板级上进行。对于一个嵌入式设备，除了处理器还要有很多周边的器件才能正常工作。因此操作系统在运行时必须初始化特定目标板的器件。这些器件中最主要的是 Flash Memory, SDRAM 等。这些设备在系统启动后必须能够正确寻址，从而使操作系统能够正常运行。此外可能需要考虑的问题包括，打印终端、串口、以太网设备等。在本文的移

植中，为了简化说明，我们只移植了少数必须的驱动程序，而网卡、MTD 等留待用户自己学习移植。

2、uClinux 内核代码基础知识

(1) 主机上的 uClinux-dist 目录结构

我们在开发时,最经常遇到的就是主机上uClinux-dist目录中的各种文件资源，而这个目录中的结构比较乱，刚上手时可能会很混乱，但理解它的目录结构，对我们开发和移植 uClinux 软件有很大的帮助,下图对其中比较重要的目录作一个说明。

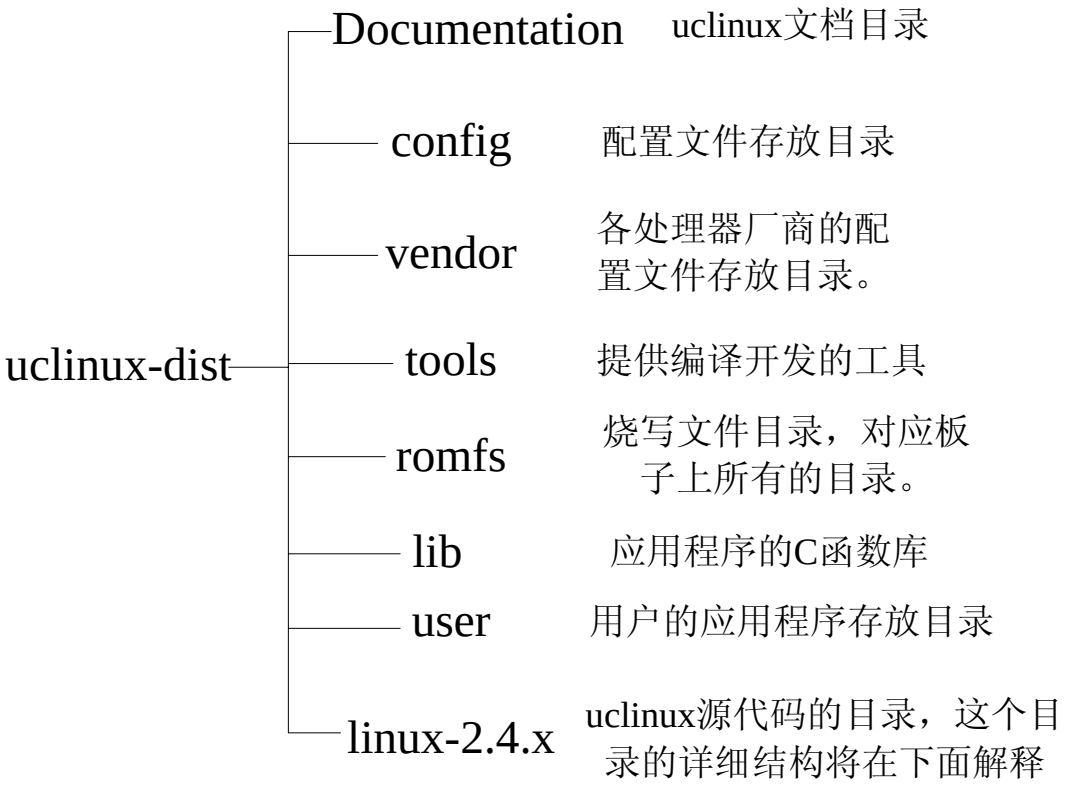


图 17-1

Documentation 目录：

这个目录下存放的是一些不同处理器平台相关的文档，但很可惜，没有 Samsung 44B0X 的相关文档，不过，其中有一篇 ARMulator-HOWTO 的文档比较值得一看，它介绍了以软件仿真的方式来调试 uClinux 内核，而不需要将我们编写的内核代码下载到目标板上进行调试，大大方便了内核软件、驱动的开发。

vendors 目录：

这个目录存放各处理器平台厂商所提供的配置文件，比如说 51EDA 的开发板的处理器采用的是三星公司的 44B0X 芯片，它所对应的配置文件存放在 vendors/Samsung/44B0X 目录下。

romfs 目录：

这个目录有点意思，你进入这个目录，然后 ls 看一下，是不是和你烧写到目标板中的目录结构一样。事实上，烧写到目标板上的映像文件，其中的文件系统 romfs 就是由这个目录所生成的。用户可以在 romfs 目录下新建一个目录，比如说

```
mkdir hello
```

然后按以下步骤编译内核

```
make image
make linux
make image
```

这时会生成新的 image.ram 和 image.rom，把新生成的映像文件烧写到目标板上，启动 uclinux 后，用 ls 指令查看目录，就可以发现新出现了一个目录 hello。而事实上，romfs 目录里是由 vendors/Samsung/44B0X/Makefile 在编译时动态生成的，因此，我们也可以修改 vendors/Samsung/44B0X/Makefile 文件来达到永久生成相应目录的目的。

user 目录：

这个目录存放的是用户自己所编写的应用程序，这个目录下的程序成功编译后，就会出现在 romfs/bin 目录下。

linux-2.4.x 目录：

这个目录存放的是 linux 的源代码，其目录结构也比较复杂，如下图如示：

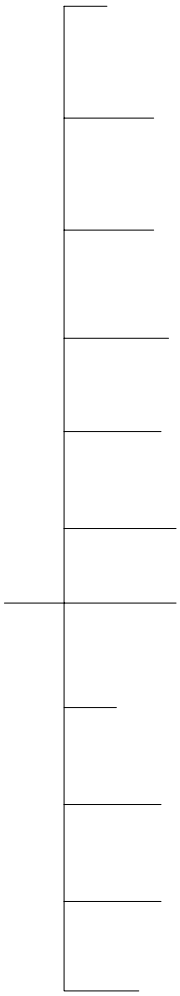


图 17-2

Documentation 目录：

此目录下是有关源代码的各种文档的集合，对程序员来说，是一笔很宝贵的财富，遇到问题时通常都能从这里找到一些思路。建议大家挑自己感兴趣的看看。

arch 目录：

在这个目录下，我们可以看到如下图所示的目录结构：

```
root@tty1[arch]# ls
alpha      cris    ia64      m68knommu    mips64      ppc      s390x  sparc64
arm        h8300   inc.txt   microblaze   niosnommu   ppc64    sh     v850
armnommu   i386    m68k      mips         parisc      s390     sparc  x86_64
```

图 17-3

其中与 arm 有关的是 arm 和 armnommu 这两项。因为 S3C44B0X 是不带 MMU(Memory

Management Unit 内存管理单元)的处理器，因此，它位于 armnommu/mach-s3c44b0x 目录下，在 armnommu/mach-s3c44b0x 目录下，有如下几个文件

```
arch.c  irq.c      Makefile      time.c
```

其中 irq.c 文件中实现了几个与中断相关的函数，如

```
void s3c44b0_mask_irq(unsigned int irq)        //屏蔽中断
void s3c44b0_unmask_irq(unsigned int irq)      //取消中断屏蔽
void s3c44b0_mask_ack_irq(unsigned int irq)    //屏蔽中断应答
void s3c44b0_int_init()         //初始化中断，包括清空中断挂号寄存器(IntPend)，使能全局中断等
```

fs 目录：

在这个目录，我们可以看到许多文件格式的源代码，包括 yaffs，ext2 等等，如果想了解某一种文件格式，查看这里的源代码可以达到事半功倍的效果。

init 目录：

系统进行各项初始化文件的存放目录，此目录下的源程序主要是处理如中断初始化、进程初始化、模块初始化和总线初始化等工作。

mm 目录和 mmnommu 目录：

这两个目录的含义是:

mm = Memory Management　　内存管理

mmnommu = Memory Management No Memory Management Unit　没有内存管理单元的内存管理，s3c44b0x 使的是这种方式。为了便于大家理解这两者的区别，下面列出了对内存管理机制的一些简单对比：

uClinux 同标准Linux 的最大区别就在于内存管理，同时也由于uClinux 的内存管理引发了一些标准Linux 所不会出现的问题。内存管理在操作系统中占有重要的地位，因此，请再读一次上文第已经谈到了Linux与uClinux内存管理机制的异同。

kernel目录：

这个目录下存放的是与内核相关的一些源代码，例如在内核调试中经常使用的printk函数就是在这个目录下实现的。

driver目录：

顾名思议，这是一个驱动程序的存放目录，这个目录下包含了种类众多的硬件驱动程序，例如USB、各种网卡、蓝牙、系统总线、摄像头、IDE、I2C等等。如果想学习linux驱动程序的编写，这个目录下的源代码是最宝贵的财富，甚至你可以从这些源代码，经过简单的移植就可以编写出符合你硬件要求的驱动程序。

include目录：

学过C语言的用户就很清楚，这个目录存放的是C头文件。这个目录下的结构如下图：

```
root@tty1[include]# ls
asm             asm-h8300       asm-mips        asm-s390        asm-x86_64  pcmcia
asm-alpha       asm-i386        asm-mips64      asm-s390x       config      scsi
asm-arm         asm-ia64        asm-niosnommu   asm-sh          linux       video
asm-armnommu    asm-m68k        asm-parisc      asm-sparc       math-emu
asm-cris        asm-m68knommu   asm-ppc         asm-sparc64     net
asm-generic     asm-microblaze  asm-ppc64       asm-v850        openssl
```

图 17-4

以asm-开头的目录是与硬件平台相关的头文件存放目录，那么编译器如何在数量众多的asm-目录中选择与我们目标板相对应的目录呢？我们可以看到在左上角有一个asm的链接，这个链接是在编译的时候，编译器根据我们目标板的硬件平台，自动建立的，用

ls asm -l

```
root@tty1[include]# ls asm -l
lrwxrwxrwx  1 root root 12 2005-01-25 20:17 asm -> asm-armnommu
```

图 17-5

我们可以看到这个asm链接事实上指向的是asm-armnommu，也就是s3c44b0x所对应的平台。

(2) make 工具和 Makefile 文件

利用make工具，我们可以将大型的开发项目分解成为多个更易于管理的模块，对于一个包括几百个源文件的应用程序，使用make和Makefile工具就可以简洁明快地理顺各个源文件之间纷繁复杂的相互关系。而且如此多的源文件，如果每次都要键入GCC命令进行编译的话，那对程序员来说简直就是一场灾难。而Make工具则可自动完成编译工作，并且可以只对程序员在上次编译后修改过的部分文件进行编译。

因此,有效的利用make和Makefile工具可以大大提高项目开发的效率。同时掌握make和Makefile之后，您也不会再面对着Linux下的应用软件手足无措了。

make工具最主要也是最基本的功能就是通过Makefile文件来描述源程序之间的相互关系并自动维护编译工作。而Makefile文件需要按照某种语法进行编写，文件中需要说明如何编译各个源文件并连接生成可执行文件，并要求定义源文件之间的依赖关系。Makefile文件是许多编译器（包括Windows NT下的编译器）维护编译信息的常用方法，只是在集成开发环境中，用户通过友好的界面修改Makefile文件而已。关于make工具的使用和Makefile的编写，可以参看附录 Make和Makefile详解。

整个uClinux各个文件合作生成一个可运行的内核映像文件的过程是通过各级目录下的Makefile文件进行组织的。因为Makefile文件的存在，使我们编译内核的工作只是需要简单的敲几个命令而已。

（3） 配置文件

编译内核所涉及的配置文件主要包括：

1. config.in

由程序 config/mkconfig 生成的文件，提供关于平台选择、程序库选择以及内核代码版本选择等的配置选项。

2. .config，autoconf.h

配置内核时产生的文件。存储着对平台和程序库以及内核代码版本选择的结果。autoconf.h 是个标准的 C 头文件，方便在代码中进行条件编译。而.config 则是下次进行配置时的默认选项。

3. config/config.in

提供编译入 Root 文件系统程序配置选项的文件。

4. config/.config，config/autoconf.h

编译入 Root 文件系统的程序的选择结果。其中，config/autoconf.h 则提供宏定义，方便进行条件编译。而 config/.config 则提供下次配置时的 默认选项。

5. linux-2.4.x/arch/armnommu/config.in

提供内核编译时的选项。添加平台等操作可以在此修改。

6. linux-2.4.x/drivers/相应的驱动程序目录/Config.in

提供内核编译时的选项，在编译内核进行选择驱动时有用。如果需要增添驱动程序，则需要更改此文件。

7. linux-2.4.x/.config，linux-2.4.x/include/linux/autoconf.h

内核部分参数的配置结果。autoconfh 为源代码中的条件编译部分提供相应的宏定义。

8. linux-2.4.x/arch/armnommu/Makefile

提供一些体系相关的编译参数，指定链接进内核镜像文件的模块，指定内核镜像文件的链接起始地址，指定平台相关的参数，调用 ld 文件链接生成最终的内核镜像文件。定义的地址参数有：TEXTADDR。

9. linux-2.4.x/arch/armnommu/vmlinux-armv.lds.in 或 vmlinux-armo.lds.in

控制链接生成内核镜像文件的 ld 程序配置文件。指定各个段的链接地址等。引用到的地址参数有：TEXTADDR。

10. linux-2.4.x/arch/armnommu/boot/Makefile

提供链接压缩内核时用到的一些地址参数定义，这些参数还将在内核解压过程中用到。定义的地址参数有：ZTEXTADDR, ZRELADDR。

11. linux-2.4.x/arch/armnommu/boot/compressed/Makefile

控制生成压缩内核文件过程，并调用 ld 程序把启动程序和内核文件链接成一个目标文件。

12. linux-2.4.x/arch/armnommu/boot/compressed/linux.lds.in

控制把启动程序和内核文件链接成一个目标文件的过程。这个文件给出了最终目标文件 zImage 中压缩过的内核的起始链接地址和末尾链接地址，input data, input-data-end.引用到的地址参数有：LOAD_ADDR，TEXT_START。

13. vendors/Samsung/44B0X/config.linux-2.4.x 或 config.vendor-2.4.x

这些文件提供了针对某一个平台的默认编译选项。

14. vendors/Samsung/44B0X/Makefile

在内核编译的过程中，将会调用此 Makefile 文件，此文件的作用是构成 root

文件系统，并进行压缩，生成 romfs.o。在生成 root 文件系统的过程中，有可能还会用到 motd、rc、inittab 等文件，则需要为此 Makefile 文件准备这些文件。平常需要增添设备，需要往 root 文件系统中增添文件都可以在此 Makefile 中进行相应的更改。

(4) 注册 Machine ID

在内核代码中，每一种设备都由一 Machine ID 来标识。所以在正式开始移植 uClinux 内核之前，需要向 uClinux 内核的维护者注册 Machine ID 和 Architecture Number，以免和别的冲突，保持向后兼容性。另外还需要在 uClinux-dist/linux-2.4.x/arch/armnommu/tools/march-types 下添加相应内容。具体修改请见下一节的内容。

这个文件在脚本 uClinux-dist/linux-2.4.x/arch/armnommu/tools/gen-mach-types 生成文件 uClinux-dist/linux-2.4.x/include/asm-armnommu/mach-types.h 时将会用到。而 mach-types.h 中定义的宏在将以后编写的代码中选择合适的代码编译进内核时用到。

(5) 内核中的 Symbol

在了解内核代码前，有很多基本的 Symbol 需要确切地了解他们的用途。下面将列出一些常用的 Symbol 进行解释。

在整个移植代码过程中，需要时刻记住实际物理地址和虚拟地址之间的映射。当内核启动起来以后，它就只处理虚拟地址，而虚拟地址和物理地址之间的映射将由体系有关的相关代码完成。这个工作由 uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/memory.h 文件中的__virt_to_phy()宏来完成。因为 44b0x 是不带 MMU 的处理器，也就是物理地址和虚拟地址相同，因此，该宏的定义很简单，没有进行转换。

```
#define        __virt_to_phy(x) ((unsigned long) (x))
```

- **解压时用到的 Symbol**

在写 uClinux 关于此平台的启动代码时，需要注意以下几个 Symbol。这几个 Symbol 在启动代码解压缩内核代码的时候将发挥重大作用。这几个 Symbol 都是在

uClinux-dist/linux-2.4.x/arch/armnommu/boot/Makefile 中定义的。

ZTEXTADDR

解压内核的解压程序的开始地址。此地址必须是实际的物理地址。

ZBSSADDR

解压程序的 BSS 段地址。必须指向 RAM 中，而且也必须是物理地址。

ZRELADDR

这是解压后，内核的开始地址。

PARAM_PHYS

启动程序给内核传递的参数的存放地址，同样为物理地址。如果没有参数传递给内核，则可以不定义。

- 内核 Symbol

PHYS_OFFSET

第一个 RAM Bank 的物理开始地址。

PAGE_OFFSET

第一个 RAM Bank 的虚拟开始地址，在没有 MMU 的处理器上，PAGE_OFFSET = PHYS_OFFSET。

TASK_SIZE

用户进程的最大字节数。用户的堆栈空间可以从此地址上逆序增长。注意不能把 IO 地址映射到比这个地址小的地址上。

TEXTADDR

内核的开始地址

DATAADDR

内核数据段的开始地址

VMALLOC_START, VMALLOC_END

vmalloc()可以分配的地址范围。在这个区域内，不能有任何静态的内存映射，再则 vmalloc()将会覆盖它。另外这段区域又必须在内核数据段内

VMALLOC_OFFSET

通常用于协助 VMALLOC_START 标识内核虚拟内存地址中的某块空的区域。

3、基于 44B0X 处理器和 51EDA 开发板的移植

在这一节的内容里，我们将从一个标准的 uClinux 出发，一步一步地完成该 uClinux 在 51EDA 开发板的移植，但该移植不包括部分驱动程序（如网卡等）。下文中只实现最基本的移植，并忽略了一些不太重要的细节，鉴于水平有限，如果移植过程有错误或是解释有错，请不吝赐教。

(1) 代码符号约定

在本节的讲解中，会大量涉及源代码，而且原来的代码和修改后的代码大量交插重叠地出发，为了避免不必要的麻烦，现将本节中用到的一些代码符号作一个约定：所有列出的源代码或代码片断都会明确在其首行给出完整路径和文件名，并且都加上了行号，如果该行没有行号，则表明此处的文字为本文添加的说明，而在代码中不会出现。原来的标准文件以斜体表示，修改后的代码通常会列在原来代码片断的后面，修改后的部分加粗显示。如下例子所示:

uClinux-dist/Makefile （文件路径）
（斜体表示原文件）

```
33   PATH   := $(PATH):$(ROOTDIR)/tools
34   HOSTCC   = unset GCC_EXEC_PREFIX; cc
35   IMAGEDIR = $(ROOTDIR)/images
36   ROMFSDIR = $(ROOTDIR)/romfs
37   ROMFSINST= romfs-inst.sh
38   SCRIPTSDIR = $(ROOTDIR)/config/scripts
39   TFTPDIR    = /tftpboot
```

uClinux-dist/ makefile（文件路径）
（非斜体表示修改后的文件）

```
33   PATH   := $(PATH):$(ROOTDIR)/tools
34   HOSTCC   = unset GCC_EXEC_PREFIX; cc
```

```
35    IMAGEDIR = $(ROOTDIR)/images
36    ROMFSDIR = $(ROOTDIR)/romfs
37    ROMFSINST= romfs-inst.sh
38    SCRIPTSDIR = $(ROOTDIR)/config/scripts
39    LINUXTARGET = bzImage        （修改的地方加粗显示）
40    TFTPDIR     = /tftpboot
```

(2) 获取 uClinux 源代码

本节所移植的代码是基于 uClinux-dist-20030305 发行版的，该文件放置在配套光盘 uClinux_porting 目录下。使用该发行版的原因是：最近发布的 uClinux 对 44B0X 的支持越来越好，这是好事，但对于我们想学习移植的用户来说，我们的工作量太小了，没有达到学习的目的，因此我们选用了较老的版本。

将该文件 uClinux-dist-20030305.tar.gz 复制到 Linux 环境中，假设将它复制到 port 目录下。

```
cp  /mnt/cdrom/uClinux_porting/uClinux-dist-20030305.tar.gz  port/
```

解压缩

```
cd port
tar  -zxf  uClinux-dist-20030305.tar.gz
```

这时，uClinux 的标准发行版源代码就已经准备好了，用 ls 指令可以看到有 uClinux-dist 目录，接下来的所有移植工作都需要在这个目录下进行。

(3) 移植实战

在这一小节中，将用一个完整的移植过程来说明移植的步骤，共有 26 步，用(a) - (z)表示。在每一步都准备无误地完成后，我们就可以得到正确的uClinux映像文件，并能看到我们熟悉的“/>”提示符。这个移植的过程只是一个抛砖引玉的过程，其中还有不少错误,欢迎大家在遇到问题或是发现错误时到www.51eda.com/bbs嵌入式操作系统版中进行讨论。

(a) uClinux-dist/Makefile

现在我们正式进入移植的工作，首先从 uClinux-dist/ Makefile 开始，这个是文件是整个 uClinux 的总领文件，它告诉编译器和连接器如何将代码组织起来。这

个文件几乎不需要更改，但为了编译后生成 image.rom 文件，我们需要加上一行代码，如下所示：

uClinux-dist/Makefile　(原文件)

```
34   PATH   := $(PATH):$(ROOTDIR)/tools
35   HOSTCC   = unset GCC_EXEC_PREFIX; cc
36   IMAGEDIR = $(ROOTDIR)/images
37   ROMFSDIR = $(ROOTDIR)/romfs
38   ROMFSINST= romfs-inst.sh
39   SCRIPTSDIR = $(ROOTDIR)/config/scripts
40   TFTPDIR    = /tftpboot
```

uClinux-dist/linux-2.4.x/makefile（修改后的文件）

```
34   PATH   := $(PATH):$(ROOTDIR)/tools
35   HOSTCC   = unset GCC_EXEC_PREFIX; cc
36   IMAGEDIR = $(ROOTDIR)/images
37   ROMFSDIR = $(ROOTDIR)/romfs
38   ROMFSINST= romfs-inst.sh
39   SCRIPTSDIR = $(ROOTDIR)/config/scripts
40   LINUXTARGET = bzImage
41   TFTPDIR    = /tftpboot
```

该文件的其余部分不需要更改。修改完该文件后，我们就要进入核心的部分，修改 linux-2.4.x 目录下的文件。而在该目录中，我们主要需要移植 arch 目录下的文件，因为这个目录下的代码是跟处理器的体系相关的，大量的汇编代码存放在该目录下。51EDA 的开发板采用的处理器为 44B0X，它是不带 MMU(Memory Manage Unit)的处理器，因此与它相关的代码存放于 uClinux-dist/linux-2.4.x/arch/armnommu 目录下。

(b) uClinux-dist/linux-2.4.x/arch/armnommu/Makefile

在该文件中加入类似如下内容：

```
ifeq ($(CONFIG_ARCH_XXX), y)
```

```
        TEXTADDR   =   0x0c008000
        MACHINE    =   xxx
    endif
```

这段代码给出了代码段的起始位置 0x0c008000 和处理器的名字 xxx，因此我们修改文件如下：

uClinux-dist/linux-2.4.x/arch/armnommu/Makefile

```
167   ifeq ($(CONFIG_BOARD_SNDS100),y)
168   TEXTADDR     = 0x00008000
169   MACHINE      = snds100
170   endif
171
172   ifeq ($(CONFIG_BOARD_EVS3C4530HEI),y)
173   TEXTADDR          = 0x00020000
174   MACHINE           = evS3C4530HEI
175   INCDIR            = $(MACHINE)
176   endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/Makefile

```
167   ifeq ($(CONFIG_BOARD_SNDS100),y)
168   TEXTADDR     = 0x00008000
169   MACHINE      = snds100
170   endif
171
172   ifeq ($(CONFIG_ARCH_S3C44B0), y)
173   TEXTADDR     = 0x0C008000
174   MACHINE      = s3c44b0
175   endif
176
177   ifeq ($(CONFIG_BOARD_EVS3C4530HEI),y)
178   TEXTADDR          = 0x00020000
179   MACHINE           = evS3C4530HEI
180   INCDIR            = $(MACHINE)
181   endif
```

(c) uClinux-dist/linux-2.4.x/arch/armnommu/config.in

为了在内核配置菜单中添加我们的处理器和开发板，我们需要修改这个文件，代码修改如下：

uClinux-dist/linux-2.4.x/arch/armnommu/config.in

```
200   if [ "$CONFIG_ARCH_SAMSUNG" = "y" ]; then
201      choice 'Board Implementation' \
202      "S3C3410-SMDK40100 CONFIG_BOARD_SMDK40100 \
203      S3C4530-HEI CONFIG_BOARD_EVS3C4530HEI \
204      S3C4510-SNDS100    CONFIG_BOARD_SNDS100" S3C4510-SNDS100
205   fi
206
207   if [ "$CONFIG_BOARD_SMDK40100" = "y" ]; then
208       define_string CONFIG_SPU_NAME "S3C3410X"
209       define_bool CONFIG_CPU_S3C3410      y
210       define_bool CONFIG_CPU_ARM710       y
211       define_bool CONFIG_CPU_32v4         y
212       define_bool CONFIG_CPU_32           y
213       define_bool CONFIG_CPU_26           n
214       define_bool CONFIG_NO_PGT_CACHE     y
215       define_bool CONFIG_CPU_WITH_CACHE   y
216       define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
217       define_int  CONFIG_ARM_CLK          40000000
218       define_bool CONFIG_SERIAL_S3C3410   y
219       if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
220          define_hex DRAM_BASE 0x00000000
221          define_hex DRAM_SIZE 0x00800000
222          define_hex FLASH_MEM_BASE 0x01000000
223          define_hex FLASH_SIZE 0x00200000
224       fi
225   fi
```

uClinux-dist/linux-2.4.x/arch/armnommu/config.in

```
200   if [ "$CONFIG_ARCH_SAMSUNG" = "y" ]; then
201      choice 'Board Implementation' \
```

```
202       "S3C3410-SMDK40100 CONFIG_BOARD_SMDK40100 \
203       S3C44B0X-51EDA CONFIG_ARCH_S3C44B0 \
          添加处理器的选择项，以便我们在内核的配置菜单中进行选择
204       S3C4530-HEI CONFIG_BOARD_EVS3C4530HEI \
205       S3C4510-SNDS100    CONFIG_BOARD_SNDS100" S3C4510-SNDS100
206   fi
定义 S3C44B0X 处理器的特性，以便于 uClinux 进行正确的编译
207   if [ "$CONFIG_ARCH_S3C44B0" = "y" ]; then
   Disable pgtable cache
208       define_bool CONFIG_NO_PGT_CACHE y
   S3C44B0X 为 32 位处理器
209       define_bool CONFIG_CPU_32 y
210       define_bool CONFIG_CPU_26 n
   S3C44B0X 为 ARM710 类型的处理器
211       define_bool CONFIG_CPU_ARM710 y
   使能 cache
212       define_bool CONFIG_CPU_WITH_CACHE y
   S3C44B0X 没有实现 MCR 指令
213       define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
   使能串口
214       define_bool CONFIG_SERIAL_S3C44B0 y
215       if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
   SDRAM 的起始地址
216           define_hex DRAM_BASE 0x0c000000
   SDRAM 的大小为 16M
217           define_hex DRAM_SIZE 0x01000000
   FLASH 的起始地址
218           define_hex FLASH_MEM_BASE 0x00000000
   FLASH 的大小
219           define_hex FLASH_SIZE 0x00200000
220       fi
221   fi
```

(d) uClinux-dist/linux-2.4.x/arch/armnommu/vmlinux-armv.lds.in

这个文件控制了 uClinux 内核映像文件的链接，比如说.text 段和.data 段的位置。在 51EDA 的开发板中，我们使用了 romfs 文件系统，因此，我们需要在这个文件中指定 romfs.o 在内核映像中的位置。

uClinux-dist/linux-2.4.x/arch/armnommu/vmlinux-armv.lds.in

```
57          __start___kallsyms = .; /* All kernel symbols           */
58              *(__kallsyms)
59          __stop___kallsyms = .;
60
61          *(.got)             /* Global offset table      */
62
63          _etext = .;         /* End of text section      */
64      }
```

uClinux-dist/linux-2.4.x/arch/armnommu/vmlinux-armv.lds.in

```
57          __start___kallsyms = .; /* All kernel symbols           */
58              *(__kallsyms)
59          __stop___kallsyms = .;
60
61          *(.got)             /* Global offset table      */
```

指定 romfs.o 在映像文件中的位置

```
62          romfs_data = .;
63          romfs.o
64          romfs_data_end = .;
65
66          _etext = .;         /* End of text section      */
67      }
```

(e) uClinux-dist/linux-2.4.x/arch/armnommu/boot/Makefile

上述文件修改完后，我们就要进入其中的子目录进行移植了，首先从 boot 目录开始，这个目录下的代码实现的功能是把内核映像解压到准确的位置，然后跳转到内核的第一句进行执行，并实现了一些初始化的工作。我们仍旧是从 Makefile 开始修改。由于我们引入了新的处理器 S3C44B0X，因此我们必须为这个处理器定位内核的段地址。文件修改如下：

uClinux-dist/linux-2.4.x/arch/armnommu/boot/Makefile

```
111    ifeq ($(CONFIG_BOARD_SNDS100),y)
112    ZRELADDR      = 0x00008000
113    ZTEXTADDR     = 0x00000000
```

```
114   endif
115
116   #
117   # If you don't define ZRELADDR above,
118   # then it defaults to ZTEXTADDR
119   #
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/Makefile

```
111   ifeq ($(CONFIG_BOARD_SNDS100),y)
112   ZRELADDR    = 0x00008000
113   ZTEXTADDR   = 0x00000000
114   endif
115
116   ifeq ($(CONFIG_ARCH_S3C44B0),y)
117   ZRELADDR    = 0x0C008000
118   ZTEXTADDR   = 0x00010000
119   ZBSSADDR    = 0x0c400000
120   endif
121
122   #
123   # If you don't define ZRELADDR above,
124   # then it defaults to ZTEXTADDR
125   #
```

(f) uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/Makefile

该文件修改如下：

```
    uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/Makefile
55   ifeq ($(CONFIG_ARCH_SA1100),y)
56   OBJS      += head-sa1100.o setup-sa1100.o
57   ifeq ($(CONFIG_SA1100_NANOENGINE),y)
58     OBJS += hw-bse.o
59   endif
60   endif
61
62   SEDFLAGS = s/TEXT_START/$(ZTEXTADDR)/;s/LOAD_ADDR/$(ZRELADDR)/;
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/Makefile

```
55   ifeq ($(CONFIG_ARCH_SA1100),y)
56   OBJS      += head-sa1100.o setup-sa1100.o
57   ifeq ($(CONFIG_SA1100_NANOENGINE),y)
58     OBJS += hw-bse.o
59   endif
60   endif
61
62   ifeq ($(CONFIG_ARCH_S3C44B0),y)
63   HEAD      = head.o
64   endif
65   SEDFLAGS = s/TEXT_START/$(ZTEXTADDR)/;s/LOAD_ADDR/$(ZRELADDR)/;
```

(g) uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/linux.lds.in

这个链接控制文件着 Bootloader 和内核映像文件的链接过程。有时还会把 root 文件系统的链接控制也放在此完成。这个文件与同目录下的 vmlinux.lds.in 文件一样，可以直接复制。

(h) uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

在些文件中，部分初始化 CPU 和各种外围器件的配置。然后跳转到 misc.c 文件中进行内核解压。此文件和 misc.c 合在一起完成 Bootloader 的功能。在编译内核的时候可以把这两个文件单独编译成一个映像文件，并还可以在这两个文件中添加一些其他的功能，这样一个 Bootloader 就完成了。这个文件被修改的部分比较多，将分成几块来说明。

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
13   #ifdef CONFIG_BOARD_SNDS100
14   #include <asm/hardware.h>
15   #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
13   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
       在本文件中用到了 SYSCFG 常量，因此需要包含 hardware.h 头文件
14   #include <asm/hardware.h>
15   #endif
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
189   #ifdef CONFIG_CPU_WITH_CACHE
190   #ifndef CONFIG_BOARD_SNDS100
191         mrc p15, 0, r6, c0, c0    @ get processor ID
192         bl  cache_on
193   #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
189   #ifdef CONFIG_CPU_WITH_CACHE
190   #ifndef CONFIG_BOARD_SNDS100
191   #ifndef CONFIG_ARCH_S3C44B0
    S3C44B0 没有实现 mrc 指令，不能执行
192         mrc p15, 0, r6, c0, c0    @ get processor ID
193         bl  cache_on
194   #endif
195   #endif
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
195   #ifdef CONFIG_BOARD_SNDS100
196         /* cache/write buffer on */
```

```
197         ldr r0, =SYSCFG
198         ldr r2, [r0]
199         orr r2, r2, #6
200         str r2, [r0]
201     #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
197   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
```

使能 cache

```
198         /* cache_on   */
199         ldr r0, =SYSCFG
200         ldr r2, [r0]
201         orr r2, r2, #6
202         str r2, [r0]
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
207     #ifndef CONFIG_BOARD_SNDS100
208         teq r4, r5          @ will we overwrite ourselves?
209         moveq   r5, r2          @ decompress after image
210         movne   r5, r4          @ decompress to final location
211     #endif
212     #ifdef CONFIG_BOARD_SNDS100
213         mov r5, r2          @ decompress after image
214     #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
209   #ifndef CONFIG_BOARD_SNDS100
210   #ifndef CONFIG_ARCH_S3C44B0
211       teq r4, r5        @ will we overwrite ourselves?
212       moveq  r5, r2        @ decompress after image
213       movne  r5, r4        @ decompress to final location
214   #endif
215   #endif
216   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
217       mov r5, r2        @ decompress after image
218   #endif
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
423       .align 5
424   cache_off:
425   #ifdef CONFIG_BOARD_SNDS100
426       ldr r0, =SYSCFG
427       ldr r1, [r0]
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
427       .align 5
428   cache_off:
429   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
430       ldr r0, =SYSCFG
431       ldr r1, [r0]
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
432   #ifndef CONFIG_BOARD_SNDS100
433   #ifdef CONFIG_CPU_ARM610
434       eor r1, r6, #0x41000000
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
436   #ifndef CONFIG_BOARD_SNDS100
437   #ifndef CONFIG_ARCH_S3C44B0
438   #ifdef CONFIG_CPU_ARM610
439         eor r1, r6, #0x41000000
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
462         mcr p15, 0, r0, c5, c0   @ invalidate whole TLB v3
463         mov pc, lr
464   #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
468         mov pc, lr
469   #endif
470   #endif
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
475         .align 5
476   cache_clean_flush:
477   #ifdef CONFIG_BOARD_SNDS100
478         ldr r1, =TAG_BASE
479         mov r2, #0
480         mov r12, #256
481
482   cache_flush_loop:
483         str r2, [r1], #4
```

```
484         subs    r12, r12, #1
485         bne cache_flush_loop
486         mov pc, lr
487     #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
481         .align 5
482     cache_clean_flush:
483     #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
484         @off_cache
485         ldr r2, =SYSCFG
    关闭 cache
486         ldr r1, =0x08
487         str r1, [r2]
488
    Cache 清 0
489         mov r1, #0
490         ldr r2, =0x10002000
491         add r12, r2, #0x2800
492     1:
493         str r1, [r2], #0x10
494         teq r2, r12
495         bne 1b
496         mov pc, lr
497     #endif
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
488     #ifndef CONFIG_BOARD_SNDS100
489         ldr r1, proc_sa110_type
490         eor r1, r1, r6
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
498   #ifndef CONFIG_BOARD_SNDS100
499   #ifndef CONFIG_ARCH_S3C44B0
500         ldr r1, proc_sa110_type
501         eor r1, r1, r6
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
577         blt 2b
578         mov pc, r10
579   #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
588         blt 2b
589         mov pc, r10
590   #endif
591   #endif
```

至此，该文件修改完毕。而且 boot 目录下的移植也完成了，下一步将进入 linux-2.4.x/arch/armnommu/kernel 目录下进行移植。

(i) uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

这个文件是 uClinux 内核启动时的第一个文件，这里体系相关的代码比较多。但理清该文件后,就可以发现,事实此文件干的工作并不多,禁止中断(IRQs 和 FIQs),将 CPU 运行在 SVC 模式，清空 BSS 段，使能 cache 然后就可以跳转到 C 的代码 start_kernel 中去了(该函数定义在 uClinux-dist/init/main.c 文件中)。

在 Bootloader 跳转到这个文件时，要满足以下条件：

- CPU 寄存器 r0＝0；
- CPU 寄存器 r1＝Machine Type ID

  (S3C44B0X 的 Machine Type ID 定义在

uClinux-dist/linux-2.4.x/include/asm-arm/mach-types.h 中：

  #define MACH_TYPE_S3C44B0 178

  寄存器 r1 也可以在 Kernel 启动之初的 head-armv.S 中设置)；
- MMU 必须关闭(S3C44B0X 没有 MMU)；
- 指令 Cache 可以打开也可以关闭，数据 Cache 必须关闭(S3C44B0X 的 Cache 是指令与数据合一的，因此只能选择关闭)。

以上条件除了第二条 r1=Machine Type ID 放在 head-armv.S 文件中处理外，其余条件都在前文中解释过的 uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S 文件中处理。请大家重新读一遍源代码，找出处理的步骤。

下面我们开始移植这个文件，同样分为分段方式来说明：

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
20   #include <asm/mach-types.h>
21   #include <asm/mach/arch.h>
22
23   #define K(a,b,c)    ((a) << 24 | (b) << 12 | (c))
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
20   #include <asm/mach-types.h>
21   #include <asm/mach/arch.h>
```

文件中用到了 SYSCFG 常量，因此加上<asm/hardware.h>头文件

```
22   #include <asm/hardware.h>
23
24   #define K(a,b,c)    ((a) << 24 | (b) << 12 | (c))
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
144   #elif defined (CONFIG_BOARD_SMDK40100)
145         mov r1, #MACH_TYPE_S3C3410
146   #endif
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
146   #elif defined (CONFIG_BOARD_SMDK40100)
147         mov r1, #MACH_TYPE_S3C3410
```

在前文已经提过，需要满足的条件之一 r1=Machine Type

```
148   #elif defined(CONFIG_ARCH_S3C44B0)
149         mov r1, #MACH_TYPE_S3C44B0
150   #endif
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
195   LC0:    .long  __bss_start
196           .long  processor_id
197           .long  _end
198           .long   __machine_arch_type
199           .long  init_task_union+8192
200   #endif
201
202   #if defined(CONFIG_BOARD_SNDS100)
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

我们需要添加一段代码，从 206 开始

```
199   LC0:        .long  __bss_start
```

```
200          .long  processor_id
201          .long  _end
202          .long   __machine_arch_type
203          .long  init_task_union+8192
204  #endif
205
206  #if defined(CONFIG_ARCH_S3C44B0)
207          adr r5, LC0
208          ldmia  r5, {r5, r6, r8, r9, sp}       @ Setup stack
209
210          /*  Clear BSS */
211          mov r4, #0
212  1:      cmp r5, r8
213          strcc  r4, [r5],#4
214          bcc 1b
215
216   /*  Pretend we know what our processor code is (for arm_id)   */
217
218          ldr r2, S3C44B0_PROCESSOR_TYPE
219
220          str     r2, [r6]
221          mov     r2, #MACH_TYPE_S3C44B0
222          str     r2, [r9]
223
224          /*  Enable Cache 8K */
225          ldr r2, =SYSCFG
   SYSCFG = 0x0e  表示使能 8K cache
226          ldr r4, =0x0e
227          str r4, [r2]
228
229          mov    fp, #0
   跳转到 uClinux-dist/linux-2.4.x/init/main.c 文件中执行
230          b   start_kernel
231
232  LC0:    .long  __bss_start
233          .long  processor_id
234          .long  _end
235          .long   __machine_arch_type
236          .long  init_task_union+8192
237
238           S3C44B0_PROCESSOR_TYPE:
```

```
239             .long  0x36366036
240     #endif
241     #if defined(CONFIG_BOARD_SNDS100)
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
233     S3C4510B_PROCESSOR_TYPE:
234      .long  0x36365036
235     #endif
236
237     #if defined(CONFIG_ARCH_SAMSUNG) && defined(CONFIG_CPU_ARM710)
238
239                     adr     r2, LC0
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
272     S3C4510B_PROCESSOR_TYPE:
273      .long  0x36365036
274     #endif
275
276     #if defined(CONFIG_ARCH_SAMSUNG) && defined(CONFIG_CPU_ARM710) &&
    !(CONFIG_ARCH_S3C44B0)
277
278                     adr     r2, LC0
```

到这步为止，这个文件的修改就完成了。

(j) uClinux-dist/linux-2.4.x/arch/armnommu/kernel/entry-armv.S

这个文件包含的是一些和平台有关的中断函数。因此，我们需要在这个文件中提供以下汇编宏：disable_fiq、get_irqnr_and_base、irq_prio_table。通常 disable_fiq 宏和 irq_prio_table 宏是空的。但是 get_irqnr_and_base 必须要小心实现。我们在这个文件的 774 行开始，加入一段 S3C44B0X 相关的代码：

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
768   1002:       /* EQ will be set if we reach 32 */
769       .endm
770
771       .macro irq_prio_table
772       .endm
773
774   #elif defined(CONFIG_ARCH_SWARM)
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/entry-armv.S

```
769   1002:       /* EQ will be set if we reach 32 */
769       .endm
770
771       .macro irq_prio_table
772       .endm
773
774   #elif defined(CONFIG_ARCH_S3C44B0)
```

disable_fiq 宏为空

```
775       .macro disable_fiq
776       .endm
777       .macro get_irqnr_and_base, irqnr, irqstat, base, tmp
778       ldr \base, =INTPND
779       ldr \base, [\base]
780       mov \irqnr, #0
781   2222:
782       tst \base, #1
783       bne 1111f
784       add \irqnr, \irqnr, #1
785       mov \base, \base, lsr #1
786       cmp \irqnr, #26
787       bcc 2222b
788   1111:
```

```
789         .endm
790
    irq_prio_table 宏也为空
791         .macro irq_prio_table
792         .endm
793
794     #elif defined(CONFIG_ARCH_SWARM)
```

(k) uClinux-dist/linux-2.4.x/arch/armnommu/kernel/irq.c

我们需要在这个文件中为 44B0X 处理器提供一个 CLEAR_PEND_INT()的函数，因此我们在该文件的 166 行加入如下代码段

```
166    #ifdef CONFIG_ARCH_S3C44B0
167     CLEAR_PEND_INT(irq);
168    #endif
```

这个文件修改完后，我们就已经完成了 kernel 目录的移植。

(l) uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

这个文件是 mm 目录下唯一需要改的文件。这个文件为 uClinux 启动时构建的 proc_info_list 结构提供数据。因此，我们修改时只需为 44B0X 处理器添加相应的数据，以便 uClinux 启动时可以从这些数据中构建 proc_info_list 结构，这个结构定义在 uClinux-dist/linux-2.4.x/include/asm-armnommu/procinfo.h 中，请大家打开这个文件研读一个这个结构，也便理解我们添加到 proc-arm6,7.S 文件中的代码。

文件分段修改如下：

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
420    cpu_s3c3410_manu_name:
421    cpu_s3c4510b_manu_name:
422    cpu_s3c4530_manu_name:
423         .asciz "Samsung"
424    cpu_s3c3410_name:
425         .asciz "S3C3410X"
426    cpu_s3c4510b_name:
427         .asciz "S3C4510B"
```

```
428   cpu_s3c4530_name:
429       .asciz  "S3C4530A01"
430       .align
```

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
420   cpu_s3c3410_manu_name:
421   cpu_s3c4510b_manu_name:
422   cpu_s3c4530_manu_name:
423   cpu_s3c44b0_manu_name:
424       .asciz "Samsung"
425   cpu_s3c3410_name:
426       .asciz "S3C3410X"
427   cpu_s3c4510b_name:
428       .asciz "S3C4510B"
429   cpu_s3c4530_name:
430       .asciz  "S3C4530A01"
431   cpu_s3c44b0_name:
432       .asciz "S3C44B0"
433       .align
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
608       .type   cpu_s3c3410_info, #object
609   cpu_s3c3410_info:
610       .long   cpu_s3c3410_manu_name
611       .long   cpu_s3c3410_name
612       .size   cpu_s3c3410_info, . - cpu_s3c3410b_info
613
614       .type   cpu_s3c4510b_info, #object
```

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
612   cpu_s3c3410_info:
613         .long   cpu_s3c3410_manu_name
614         .long   cpu_s3c3410_name
615         .size   cpu_s3c3410_info, . - cpu_s3c3410b_info
616
```

加入 44B0X 相关的代码

```
617         .type   cpu_s3c44b0_info, #object
618   cpu_s3c44b0_info:
619         .long    cpu_s3c44b0_manu_name
620         .long    cpu_s3c44b0_name
621         .size   cpu_s3c44b0_info, . - cpu_s3c44b0_info
622         .type   cpu_s3c4510b_info, #object
```

---

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
722   __s3c4510b_proc_info:
723                 .long   0x36365000              @ cpu_val
724                 .long   0xfffff000              @ cpu_mask
725                 .long   0x00000c1e              @ __cpu_mmu_flags
726                 b       __arm7_setup            @ __cpu_flush
727                 .long   cpu_arch_name           @ arch_name
728                 .long   cpu_elf_name                    @ elf_name
729                 .long   HWCAP_SWP | HWCAP_26BIT         @ elf_hwcap
730                 .long   cpu_s3c4510b_info               @ info
731                 .long   arm7_processor_functions        @ info
732               .size   __s3c4510b_proc_info, . - __s3c4510b_proc_info
733
734                 .type   __s3c4530_proc_info, #object
```

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
731   __s3c4510b_proc_info:
732                   .long   0x36365000                      @ cpu_val
733                   .long   0xfffff000                      @ cpu_mask
734                   .long   0x00000c1e              @ __cpu_mmu_flags
735                   b       __arm7_setup              @ __cpu_flush
736                   .long   cpu_arch_name                 @ arch_name
737                   .long   cpu_elf_name                    @ elf_name
738                   .long   HWCAP_SWP | HWCAP_26BIT        @ elf_hwcap
739                   .long   cpu_s3c4510b_info               @ info
740                   .long   arm7_processor_functions        @ info
741               .size   __s3c4510b_proc_info, . - __s3c4510b_proc_info
742
743                   .type __s3c44b0_proc_info, #object
```

加入 s3c44b0 相关的代码

```
744   __s3c44b0_proc_info:
745                   .long  0x36366000
746                   .long  0xFFFFF000
747                   .long  0x00000c1e
748                   b   __arm7_setup
749                   .long  cpu_arch_name
750                   .long  cpu_elf_name
751                   .long  HWCAP_SWP | HWCAP_26BIT
752                   .long  cpu_s3c44b0_info
753                   .long  arm7_processor_functions
754                   .size  __s3c44b0_proc_info, . - __s3c44b0_proc_info
755
756                   .type  __s3c4530_proc_info, #object
```

这个文件移植完成。

(m) mach-s3c44b0 目录

在 uClinux-dist/linux-2.4.x/arch/armnommu 目录下有很多以“mach-”开头的目录，这些都是跟体系相关的一些代码，我们需要为我们的 S3C44B0 处理器也建立一个目录，用以存放相关代码。那在编译时，编译器如何知道我们需要用哪一个

目录下的文件呢？在 uClinux-dist/linux-2.4.x/arch/armnommu/Makefile 中，已经指明了用以参与编译的目录，相关代码如下，其中加粗的代码行说明了我们编译所需用的目录为 mach-s3c44b0。

uClinux-dist/linux-2.4.x/arch/armnommu/Makefile

```
196   # If we have a machine-specific directory, then include it in the build.
197   MACHDIR     := arch/armnommu/mach-$(MACHINE)
198   ifeq ($(MACHDIR),$(wildcard $(MACHDIR)))
199   SUBDIRS     += $(MACHDIR)
200   CORE_FILES  := $(MACHDIR)/$(MACHINE).o $(CORE_FILES)
201   endif
```

(n) uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/Makefile

Makefile 的作用不需要重复了，源代码如下：

```
1   #
2   # Makefile for the linux kernel.
3   #
4   # Note! Dependencies are done automagically by 'make dep', which also
5   # removes any old dependencies. DON'T put your own dependencies here
6   # unless it's something special (ie not a .c file).
7
8   USE_STANDARD_AS_RULE := true
9
10   O_TARGET     := s3c44b0.o
11
12   # Object file lists.
13
14   obj-y        := $(patsubst %.c, %.o, $(wildcard *.c))
15   obj-m        :=
16   obj-n        :=
17   obj-         :=
18
19   export-objs     :=
20
21   include $(TOPDIR)/Rules.make
```

(o) uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/arch.c

该文件里包含的是一些体系相关的 IO 初始化代码。代码如下：

uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/arch.c

```
1   /*
2    *  linux/arch/arm/mach-s3c44b0/arch.c
3    *
4    *  Architecture specific fixups.  This is where any
5    *  parameters in the params struct are fixed up, or
6    *  any additional architecture specific information
7    *  is pulled from the params struct.
8    */
9   #include <linux/tty.h>
10   #include <linux/delay.h>
11   #include <linux/pm.h>
12   #include <linux/init.h>
13
14   #include <asm/elf.h>
15   #include <asm/setup.h>
16   #include <asm/mach-types.h>
17   #include <asm/mach/arch.h>
18
19   extern void genarch_init_irq(void);
20
21   MACHINE_START(S3C44B0, "51EDA")
22          MAINTAINER("Lei Ni")
23          BOOT_MEM(DRAM_BASE,0x00000000,0x00000000)
24          INITIRQ(genarch_init_irq)
25   MACHINE_END
```

(p) uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/irq.c

提供以下函数的定义

void s3c44b0_mask_irq(unsigned int irq)

void s3c44b0_unmask_irq(unsigned int irq)

void s3c44b0_mask_ack_irq(unsigned int irq)

void s3c44b0_int_init(unsigned int irq)

以上函数的实现都比较容易，代码如下：

uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/irq.c

```
1   /*
```

```
2   *  linux/arch/armnommu/mach-s3c44b0/irq.c
3   *  2001 Mac Wang <mac@os.nctu.edu.tw>
4   */
5   #include <linux/init.h>
6
7   #include <asm/mach/irq.h>
8   #include <asm/hardware.h>
9   #include <asm/io.h>
10   #include <asm/irq.h>
11   #include <asm/system.h>
12
13   void s3c44b0_mask_irq(unsigned int irq)
14   {
15     INT_DISABLE(irq);
16   }
17
18   void s3c44b0_unmask_irq(unsigned int irq)
19   {
20     INT_ENABLE(irq);
21   }
22
23   void s3c44b0_mask_ack_irq(unsigned int irq)
24   {
25     INT_DISABLE(irq);
26   }
27
28   void s3c44b0_int_init()
29   {
30     IntPend = 0x1FFFFF;
31     IntMode = INT_MODE_IRQ;
32     INT_ENABLE(INT_GLOBAL);
33     *(unsigned int *)INTCON = 5;
34   }
```

(q) uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/irq.c

```
1   /*
2    * time.c  Timer functions for Samsung 44B0
3    */
4
5   #include <linux/time.h>
```

```
6   #include <linux/timex.h>
7   #include <linux/types.h>
8   #include <linux/sched.h>
9   #include <asm/io.h>
10   #include <asm/arch/hardware.h>
11
12   unsigned long samsung_gettimeoffset (void)
13   {
14     return 0;
15   }
16
17   void samsung_timer_interrupt(int irq, void *dev_id, struct pt_regs
*regs)
18   {
20     do_timer(regs);
21   }
```

(r) uClinux-dist/linux-2.4.x/arch/armnommu/tools/mach-types

在上一小节中，已经说明在移植内核时，需要注册 Machine ID，通过修改这个文件来实现，我们在这个文件中添加 S3C44B0 的 Machine ID，源代码如下：

```
98   # The following are unallocated
99
100   p52             ARCH_P52              P52           87
101   spipe           ARCH_SPIPE            SPIPE         88
102   atmel           ARCH_ATMEL            ATMEL         89
103   dsc21           ARCH_DSC21            DSC21         115
104   snds100         ARCH_SNDS100          SNDS100       90
105   s3c44b0         ARCH_S3C44B0          S3C44B0       178
106   evS3C4530HEI    BOARD_EVS3C4530HEI    EVS3C4530HEI  164
107   S3C3410X        BOARD_SMDK40100       S3C3410       165
```

(s) 头文件移植

在内核程序中,经常会用到中断操作或是对S3C44B0X处理器的寄存器进行操作，因此我们需要移植这部分体系相关的头文件。移植的工作如下：

- 创建 uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0 目录
- 在上述目录中编写以下文件：

dma.h
hardware.h
io.h
irq.h
irqs.h
memory.h
param.h
processor.h
serial.h
system.h
time.h
timex.h
uncompress.c
uncompress.h
vmalloc.h

这些文件的源代码都可以在下一节代码列表中查到，值得注意的是，这些头文件中并没有定义所有的寄存器，只是定义了我们移植时需要用到的，而 DMA、IIS 相关的寄存器都没有在 hardware.h 文件中定义，如果有用到，请自行修改。

- 修改 proc-armv/system.h 文件

uClinux-dist/linux-2.4.x/include/asm-armnommu/proc-armv/system.h

```
39 #ifdef __ARM_ARCH_4__
40 #define vectors_base()   ((cr_alignment & CR_V) ? 0xffff0000 : 0)
41 #else
42 #define vectors_base()   (0)
43 #endif
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/proc-armv/system.h

```
39 #ifdef CONFIG_ARCH_S3C44B0
40 #define vectors_base()   (0x0C000000)
41 #elif
42 #define    vectors_base()    (0)
43 #endif
```

(t) 驱动程序移植

为了正确启动，我们至少需要移植串口和 blockmem 两部分驱动。串口的移植需要修改 uClinux-dist/linux-2.4.x/drivers/char/目录下的以下三个文件：

Config.in
Makefile
tty_io.c

并需要编写以下四个新的文件

s3c44b0.c
s3c44b0.h
serial_core_44b0x.c
serial_core_44b0x.h

而 blockmem 的移植仅需要修改 uClinux-dist/linux-2.4.x/drivers/block/blkmem.c 文件。下面将分别说明移植的步骤。

(u) uClinux-dist/linux-2.4.x/drivers/char/Config.in

我们需要在这个文件中加入 S3C44B0X 串口的信息，以便可以在内核配置中选取串口。在该文件的 114 行加入以下代码：

```
114   if [ "$CONFIG_ARCH_S3C44B0" = "y" ]; then
115      bool 'S3C44B0 serial port support' CONFIG_SERIAL_S3C44B0
116      if [ "$CONFIG_SERIAL_S3C44B0" = 'y' ]; then
117               bool ' Support for console on S3C44B0 serial port'
CONFIG_SERIAL_S3C44B0_CONSOLE
118      fi
119   fi
```

(v) uClinux-dist/linux-2.4.x/drivers/char/Makefile

修改 Makefile 文件，以便把 s3c44b0x.c 和 serial_core_44b0x.c 文件编译到内核中。修改如下：

uClinux-dist/linux-2.4.x/drivers/char/Makefile

```
224   obj-$(CONFIG_SERIAL_NETARM) += serial_netarm.o
225   obj-$(CONFIG_SERIAL_SAMSUNG) += serial_samsung.o
226   obj-$(CONFIG_SERIAL_S3C3410) += serial_s3c3410.o
227   obj-$(CONFIG_SERIAL_S3C4530) += serial_s3c4530.o
```

uClinux-dist/linux-2.4.x/drivers/char/Makefile

```
224   obj-$(CONFIG_SERIAL_NETARM) += serial_netarm.o
225   obj-$(CONFIG_SERIAL_SAMSUNG) += serial_samsung.o
226   obj-$(CONFIG_SERIAL_S3C44B0) += serial_core_44b0x.o
227   obj-$(CONFIG_SERIAL_S3C44B0) += s3c44b0.o
228   obj-$(CONFIG_SERIAL_S3C3410) += serial_s3c3410.o
229   obj-$(CONFIG_SERIAL_S3C4530) += serial_s3c4530.o
```

(w) uClinux-dist/linux-2.4.x/drivers/char/tty_io.c

这个文件的修改很简单，修改如下：

uClinux-dist/linux-2.4.x/drivers/char/tty_io.c

```
2287   #ifdef CONFIG_SERIAL_SAMSUNG_CONSOLE
2288       samsung_console_init();
2289   #endif
2290   #ifdef CONFIG_SREIAL_S3C4530_CONSOLE
2291       s3c4530_console_init();
2292   #endif
```

uClinux-dist/linux-2.4.x/drivers/char/tty_io.c

```
2287   #ifdef CONFIG_SERIAL_SAMSUNG_CONSOLE
2288       samsung_console_init();
2289   #endif
2290   #ifdef CONFIG_SERIAL_S3C44B0_CONSOLE
2291       samsung_console_init();
2292   #endif
2293   #ifdef CONFIG_SREIAL_S3C4530_CONSOLE
2294       s3c4530_console_init();
```

```
2295   #endif
```

(x) 在 uClinux-dist/linux-2.4.x/drivers/char 目录下添加四个文件

这四个文件分别为：s3c44b0x.h、s3c44b0x.c、serial_core_44b0x.h、serial_core_44b0x.c，这四个文件完成了 44B0X 处理器的串口驱动任务，源代码就查看下一小节--文件清单。

(y) uClinux-dist/linux-2.4.x/drivers/block/blkmem.c

对这个文件的修改主要是与 romfs 文件系统有关，修改过程也比较简单：

uClinux-dist/linux-2.4.x/drivers/block/blkmem.c

```
79   /*
80    * Please, configure the ROMFS for your system here
81    */
82
83   /* v850e; this config stuff is ugly, ugly, ugly! */
84   #ifdef CONFIG_V850E
```

uClinux-dist/linux-2.4.x/drivers/block/blkmem.c

```
79   /*
80    * Please, configure the ROMFS for your system here
81    */
82
83   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0) ||
defined(CONFIG_ARCH_INTEGRATOR)
84   extern char romfs_data[];
85   extern char romfs_data_end[];
86   #endif
87   /* v850e; this config stuff is ugly, ugly, ugly! */
88   #ifdef CONFIG_V850E
```

uClinux-dist/linux-2.4.x/drivers/block/blkmem.c

```
344   #ifdef FIXED_ROMARRAY
345    {0, (unsigned long) FIXED_ROMARRAY, -1},
346   #endif
347   #ifdef CONFIG_ARCH_CNXT
348       /*  AM29LV004T flash
```

uClinux-dist/linux-2.4.x/drivers/block/blkmem.c

```
348   #ifdef FIXED_ROMARRAY
349    {0, (unsigned long) FIXED_ROMARRAY, -1},
350   #endif
351
352   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
|| defined(CONFIG_ARCH_INTEGRATOR)
353    {0, romfs_data, -1},
354   #endif
355
356   #ifdef CONFIG_ARCH_CNXT
357       /*  AM29LV004T flash
```

移植到这步时，uClinux-dist/linux-2.4.x 目录下的工作已经结束，下面要在 uClinux-dist/vendor 目录下做文章。

(z) uClinux-dist/vendor/Samsung/44B0X

首先要在 uClinux-dist/vendor/Samsung 目录下添加 44B0X 目录，并在这个目录下添加四个文件，分别为

Makefile ：这个文件的作用不需多说了吧，控制编译过程

config.arch ：配置文件

motd ：uClinux 启动后的 logo，可以执行 cat motd 指令来看看输出

rc ：这个文件让你的 uClinux 启动后自己执行其中的指令。

具体的源代码请看下一节--文件清单。

(4) 编译 uClinux

完成移植工作后，回到 uClinux-dist 目录下，按以下步骤编译 uClinux，如果遇到错误，请大家仔细排察错误。因为现在这个内核是一清二白的，因此我们需要针对我们的开发板进行相应的设置，下面以 51EDA 的开发板为例进行说明。

make menuconfig

首先要对内核进行配置，执行该条指令后，会有很多提示信息，这时一律按回车，直到出现以下界面：

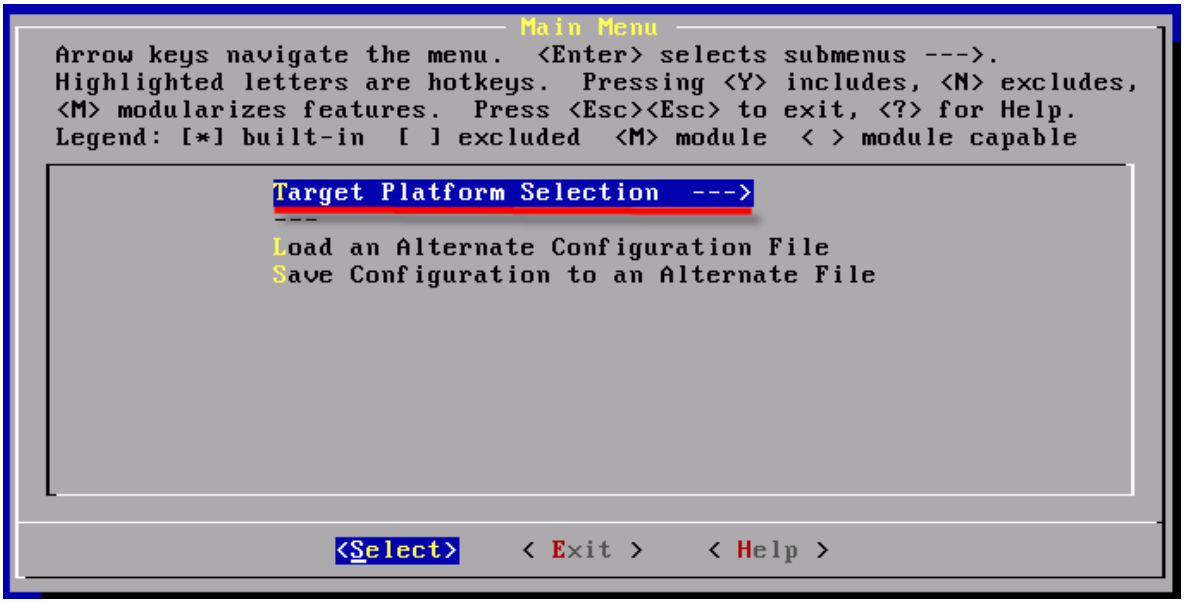


图 17-6

在上图红线处回车，进入下一个配置界面，如下图所示：

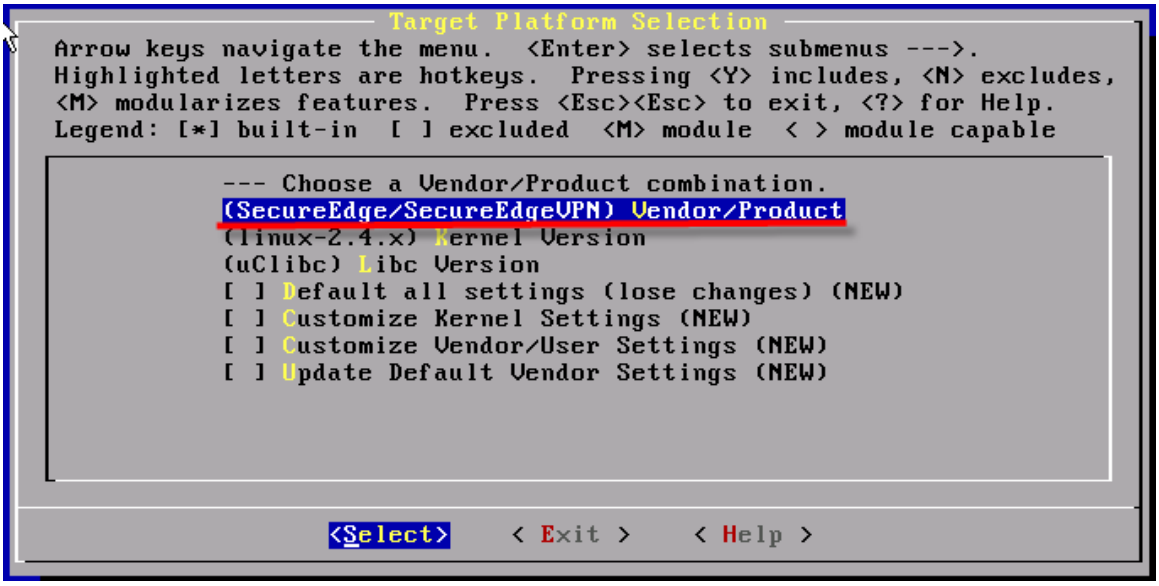


图 17-7

同样在红线处回车，以进行处理器和平台的选择，如下图：

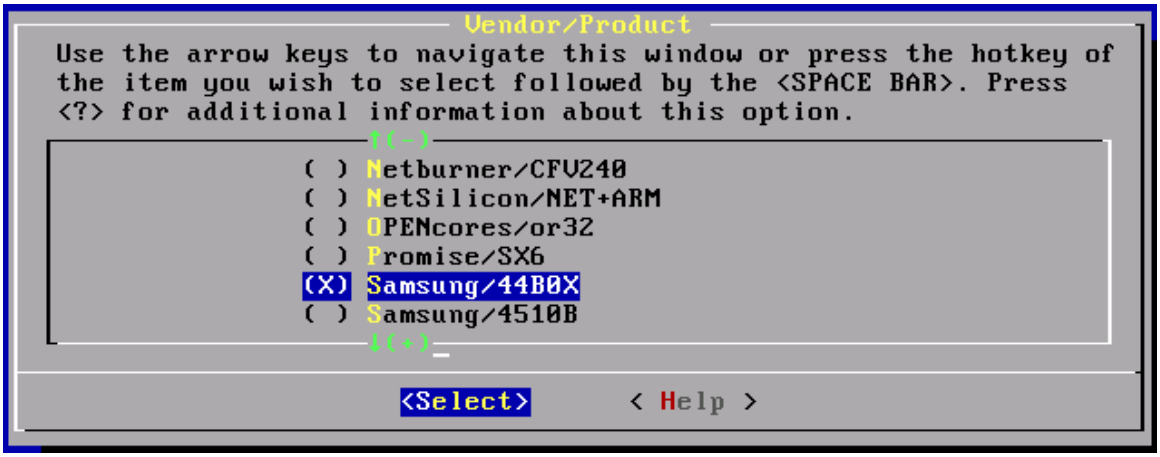


图 17-8

在蓝色光标的位置按空格选中 Samsung/44B0X 处理器，并回到下图的配置界面：

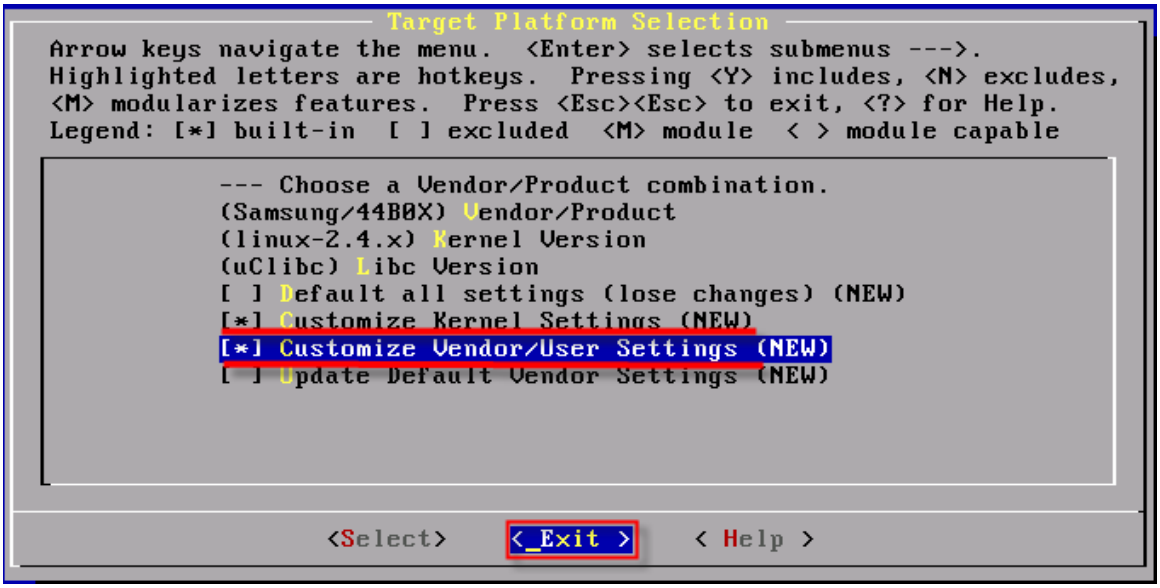


图 17-9

用空格选中上图红线部分，并两次选择“Exit”回车，当出现如下图所示的提示对话框时，选择“Yes”。

图 17-10

进入新的配置界面，如下图：

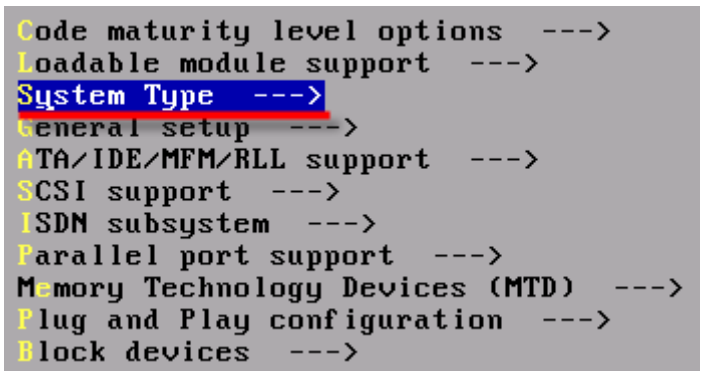


图 17-11

选择上图红线处的选项，进行平台的选择，如下图：

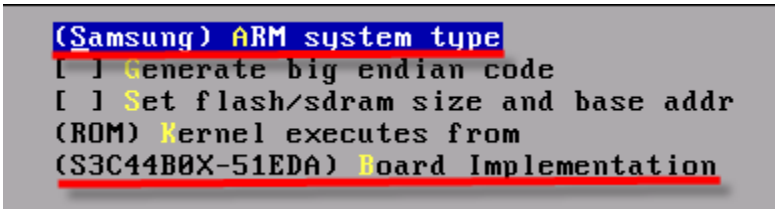


图 17-12

将上图红线处的两个选项按上图的要求进行设置，然后返回上一级界面。

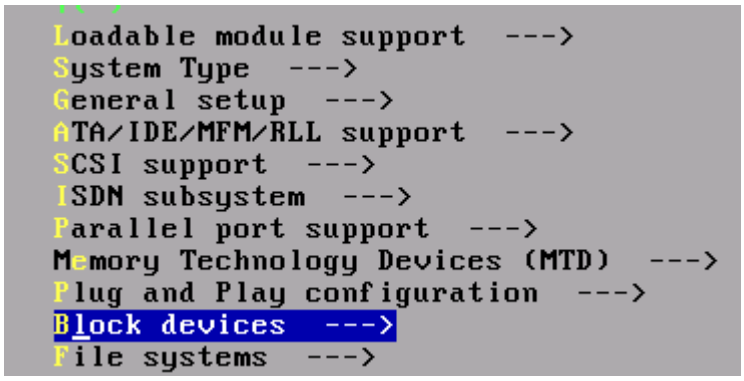


图 17-13

选中 Block devices，进入下一级配置界面：

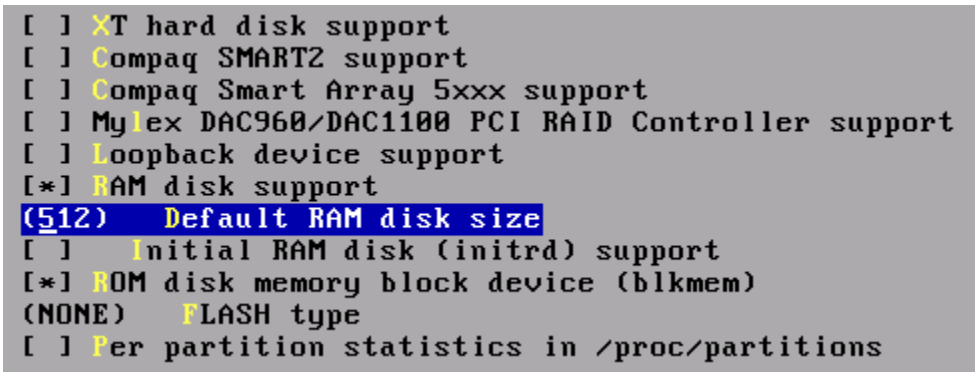


图 17-14

按上图的要求进行设置，并返回上一级配置界面：

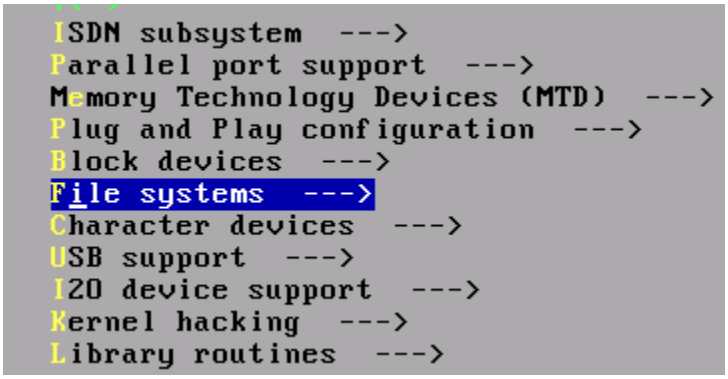


图 17-15

选中 File systems，进入下一级设置菜单

图 17-16

设置见上图所示，然后再返回上一级菜单：

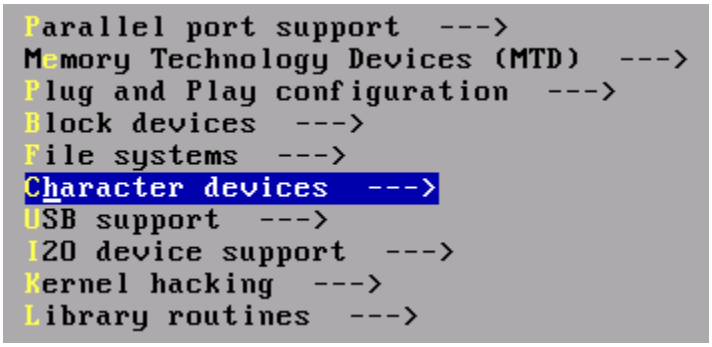


图 17-17

这回选中 Character devices，主要是配置串口，如下图所示：

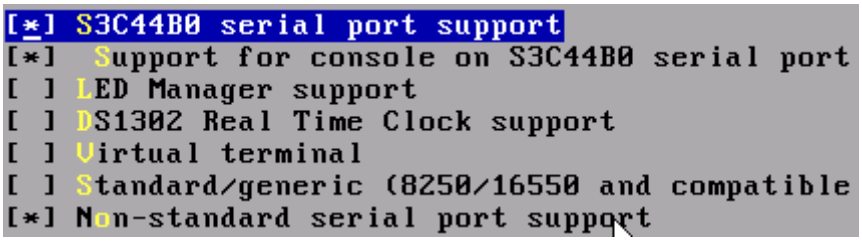


图 17-18

设置结果如上图所示，这时再返回上一级菜单：

图 17-19

再选择“Exit”退出，在弹出的对话框时选择“Yes”保存内核设置。稍等一会，我们将进入另一个新的配置界面，这个配置界面主要是应用程序的设置。

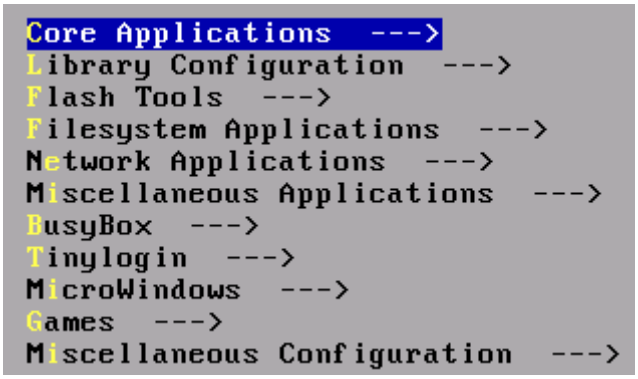


图 17-20

这时选择上图中的“Core Applications”，进入下一级界面：

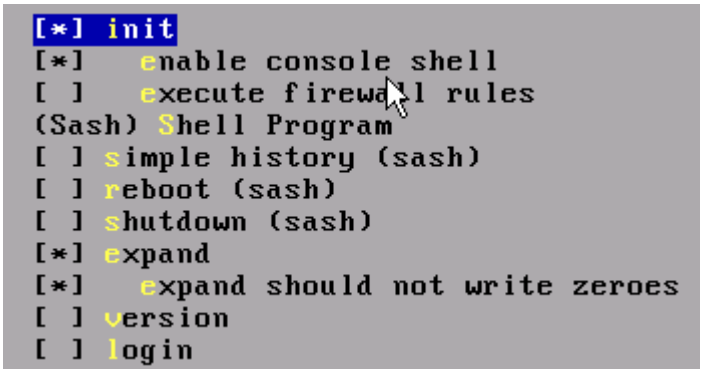


图 17-21

按上图的要求进行设置，然后两次选择“Exit”，并在提示对话框处选择“Yes”，保存内核设置，这样，我们的全部设置工作就完成了。再按下面的指令，按顺序进行编译。

```
make  dep
make  lib_only
make  user_only
make  romfs
make  image (第一次执行时会有错，不必理会)
make  linux
make  image
```

如果整个过程都没有错误，我们可以在 uClinux-dist/image 目录下找到两个映像文件 image.ram 和 image.rom，然后按前文的说明，把 image.rom 烧到到开发板的 Flash 中，如果你看到了如下图所示的 uClinux logo 和\>提示符，那恭喜你，内核移植成功。

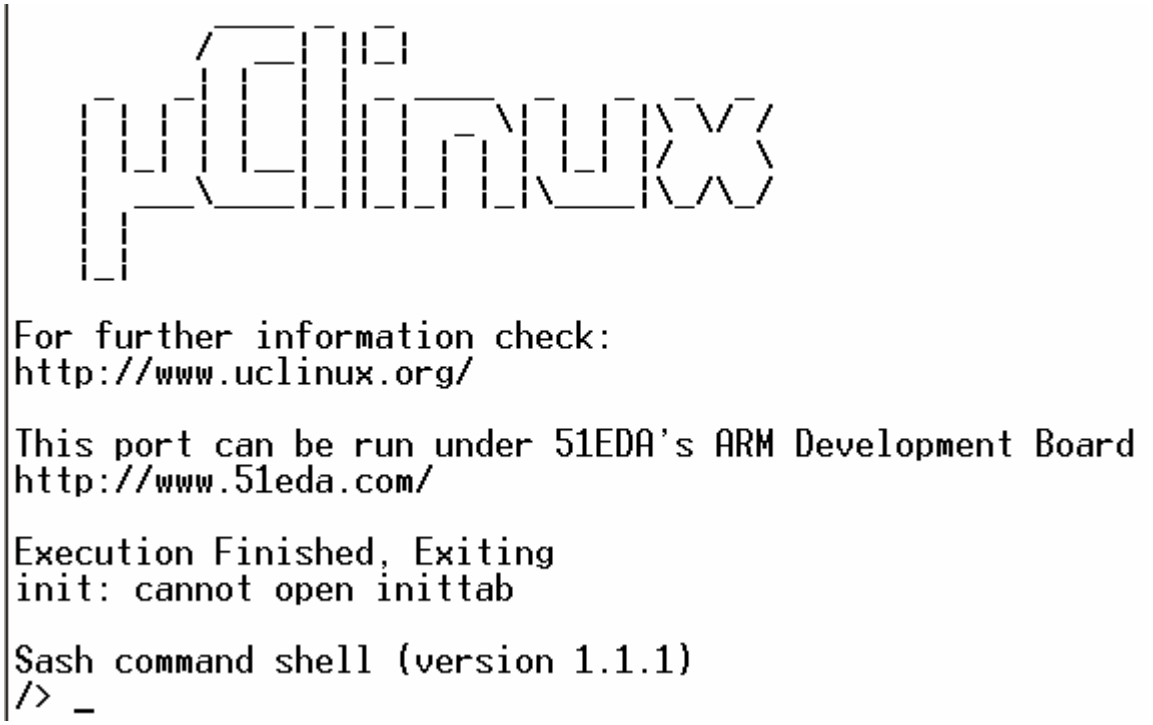


图 17-22

(4) 文件清单

在本小节里，我们会把移植过程中改动较大的或添加过的文件全部列出，并加上行号，以方便大家查阅。并且，所有被修改或改动过的文件都在配套光盘\port_list 目录下，按目录结构的方式给出。

uClinux-dist/Makefile

```
1   #############################################################################
2
3   #
4   # Makefile -- Top level uClinux makefile.
5   #
6   # Copyright (c) 2001-2002, SnapGear (www.snapgear.com)
7   # Copyright (c) 2001, Lineo
8   #
9
10   VERSIONPKG = 1.3.4
11   VERSIONSTR = $(CONFIG_VENDOR)/$(CONFIG_PRODUCT) Version $(VERSIONPKG)
12
13   #############################################################################
14   #
15   # Lets work out what the user wants, and if they have configured us yet
16   #
17
18   ifeq (.config,$(wildcard .config))
19   include .config
20
21   all: subdirs romfs modules modules_install image
22   else
23   all: config_error
24   endif
25
26   #############################################################################
27   #
28   # Get the core stuff worked out
29   #
30
31   LINUXDIR = $(CONFIG_LINUXDIR)
32   LIBCDIR  = $(CONFIG_LIBCDIR)
33   ROOTDIR  = $(shell pwd)
34   PATH    := $(PATH):$(ROOTDIR)/tools
35   HOSTCC   = unset GCC_EXEC_PREFIX; cc
36   IMAGEDIR = $(ROOTDIR)/images
37   ROMFSDIR = $(ROOTDIR)/romfs
38   ROMFSINST= romfs-inst.sh
39   SCRIPTSDIR = $(ROOTDIR)/config/scripts
40   LINUXTARGET = bzImage
41   TFTPDIR    = /tftpboot
42
43   LINUX_CONFIG  = $(ROOTDIR)/$(LINUXDIR)/.config
44   CONFIG_CONFIG = $(ROOTDIR)/config/.config
45   MODULES_CONFIG = $(ROOTDIR)/modules/.config
```

```
46
47  CONFIG_SHELL := $(shell if [ -x "$$BASH" ]; then echo $$BASH; \
48       else if [ -x /bin/bash ]; then echo /bin/bash; \
49       else echo sh; fi ; fi)
50
51  ifeq (config.arch,$(wildcard config.arch))
52  include config.arch
53  ARCH_CONFIG = $(ROOTDIR)/config.arch
54  export ARCH_CONFIG
55  endif
56
57  ifneq ($(SUBARCH),)
58  # Using UML, so make the kernel and non-kernel with different ARCHs
59  MAKEARCH = $(MAKE) ARCH=$(SUBARCH) CROSS_COMPILE=$(CROSS_COMPILE)
60           MAKEARCH_KERNEL    =    $(MAKE)    ARCH=$(ARCH)    SUBARCH=$(SUBARCH)
CROSS_COMPILE=$(CROSS_COMPILE)
61  else
62  MAKEARCH = $(MAKE) ARCH=$(ARCH) CROSS_COMPILE=$(CROSS_COMPILE)
63  MAKEARCH_KERNEL = $(MAKEARCH)
64  endif
65
66
67  DIRS    = $(VENDOR_TOPDIRS) lib user
68  VENDDIR = $(ROOTDIR)/vendors/$(CONFIG_VENDOR)/$(CONFIG_PRODUCT)/.
69
70  export VENDOR PRODUCT ROOTDIR LINUXDIR HOSTCC CONFIG_SHELL
71  export CONFIG_CONFIG LINUX_CONFIG ROMFSDIR SCRIPTSDIR
72  export VERSIONPKG VERSIONSTR ROMFSINST PATH IMAGEDIR TFTPDIR
73
74  ###############################################################################
75
76  #
77  # Config stuff,  we recall ourselves to load the new config.arch before
78  # running the kernel and other config scripts
79  #
80
81  .PHONY: config.tk config.in
82
83  config.in:
84     config/mkconfig > config.in
85
86  config.tk: config.in
87     $(MAKE) -C $(SCRIPTSDIR) tkparse
88     ARCH=dummy $(SCRIPTSDIR)/tkparse < config.in > config.tmp
89     @if [ -f /usr/local/bin/wish ]; then \
90         echo '#!'"/usr/local/bin/wish -f" > config.tk; \
91     else \
```

```
92          echo '#!'"/usr/bin/wish -f" > config.tk; \
93      fi
94      cat $(SCRIPTSDIR)/header.tk >> ./config.tk
95      cat config.tmp >> config.tk
96      rm -f config.tmp
97      echo "set defaults \"/dev/null\"" >> config.tk
98      echo "set help_file \"config/Configure.help\"" >> config.tk
99      cat $(SCRIPTSDIR)/tail.tk >> config.tk
100     chmod 755 config.tk
101
102   .PHONY: xconfig
103   xconfig: config.tk
104     @wish -f config.tk
105     @if [ ! -f .config ]; then \
106         echo; \
107         echo "You have not saved your config, please re-run make config"; \
108         echo; \
109         exit 1; \
110      fi
111     @config/setconfig defaults
112     @if egrep "^CONFIG_DEFAULTS_KERNEL=y" .config > /dev/null; then \
113         $(MAKE) linux_xconfig; \
114      fi
115     @if egrep "^CONFIG_DEFAULTS_MODULES=y" .config > /dev/null; then \
116         $(MAKE) modules_xconfig; \
117      fi
118     @if egrep "^CONFIG_DEFAULTS_VENDOR=y" .config > /dev/null; then \
119         $(MAKE) config_xconfig; \
120      fi
121     @config/setconfig final
122
123   .PHONY: config
124   config: config.in
125     @HELP_FILE=config/Configure.help \
126         $(CONFIG_SHELL) $(SCRIPTSDIR)/Configure config.in
127     @config/setconfig defaults
128     @if egrep "^CONFIG_DEFAULTS_KERNEL=y" .config > /dev/null; then \
129         $(MAKE) linux_config; \
130      fi
131     @if egrep "^CONFIG_DEFAULTS_MODULES=y" .config > /dev/null; then \
132         $(MAKE) modules_config; \
133      fi
134     @if egrep "^CONFIG_DEFAULTS_VENDOR=y" .config > /dev/null; then \
135         $(MAKE) config_config; \
136      fi
137     @config/setconfig final
138
```

```
139  .PHONY: menuconfig
140  menuconfig: config.in
141    $(MAKE) -C $(SCRIPTSDIR)/lxdialog all
142    @HELP_FILE=config/Configure.help \
143        $(CONFIG_SHELL) $(SCRIPTSDIR)/Menuconfig config.in
144    @if [ ! -f .config ]; then \
145        echo; \
146        echo "You have not saved your config, please re-run make config"; \
147        echo; \
148        exit 1; \
149     fi
150    @config/setconfig defaults
151    @if egrep "^CONFIG_DEFAULTS_KERNEL=y" .config > /dev/null; then \
152        $(MAKE) linux_menuconfig; \
153     fi
154    @if egrep "^CONFIG_DEFAULTS_MODULES=y" .config > /dev/null; then \
155        $(MAKE) modules_menuconfig; \
156     fi
157    @if egrep "^CONFIG_DEFAULTS_VENDOR=y" .config > /dev/null; then \
158        $(MAKE) config_menuconfig; \
159     fi
160    @config/setconfig final
161
162  .PHONY: oldconfig
163  oldconfig:
164    @$(MAKE) oldconfig_linux
165    @$(MAKE) oldconfig_modules
166    @$(MAKE) oldconfig_config
167    @config/setconfig final
168
169  .PHONY: modules
170  modules:
171    . $(LINUXDIR)/.config; if [ "$$CONFIG_MODULES" = "y" ]; then \
172        [ -d $(LINUXDIR)/modules ] || mkdir $(LINUXDIR)/modules; \
173        $(MAKEARCH_KERNEL) -C $(LINUXDIR) modules; \
174    fi
175
176  .PHONY: modules_install
177  modules_install:
178    . $(LINUXDIR)/.config; if [ "$$CONFIG_MODULES" = "y" ]; then \
179        [ -d $(ROMFSDIR)/lib/modules ] || mkdir -p $(ROMFSDIR)/lib/modules; \
180        $(MAKEARCH_KERNEL)    -C    $(LINUXDIR)    INSTALL_MOD_PATH=$(ROMFSDIR)
DEPMOD=true modules_install; \
181        rm -f $(ROMFSDIR)/lib/modules/*/build; \
182        find $(ROMFSDIR)/lib/modules -type f | xargs -r $(STRIP) -g; \
183    fi
184
```

```
185  linux_xconfig:
186    $(MAKEARCH_KERNEL) -C $(LINUXDIR) xconfig
187  linux_menuconfig:
188    $(MAKEARCH_KERNEL) -C $(LINUXDIR) menuconfig
189  linux_config:
190    $(MAKEARCH_KERNEL) -C $(LINUXDIR) config
191  modules_xconfig:
192    [ ! -d modules ] || $(MAKEARCH) -C modules xconfig
193  modules_menuconfig:
194    [ ! -d modules ] || $(MAKEARCH) -C modules menuconfig
195  modules_config:
196    [ ! -d modules ] || $(MAKEARCH) -C modules config
197  modules_clean:
198    -[ ! -d modules ] || $(MAKEARCH) -C modules clean
199  config_xconfig:
200    $(MAKEARCH) -C config xconfig
201  config_menuconfig:
202    $(MAKEARCH) -C config menuconfig
203  config_config:
204    $(MAKEARCH) -C config config
205  oldconfig_config:
206    $(MAKEARCH) -C config oldconfig
207  oldconfig_modules:
208    [ ! -d modules ] || $(MAKEARCH) -C modules oldconfig
209  oldconfig_linux:
210    $(MAKEARCH_KERNEL) -C $(LINUXDIR) oldconfig
211
212  #########################################################################################
213  #
214  # normal make targets
215  #
216
217  .PHONY: romfs
218  romfs:
219    for dir in $(DIRS) ; do $(MAKEARCH) -C $$dir romfs || exit 1 ; done
220    -find $(ROMFSDIR)/. -name CVS | xargs -r rm -rf
221
222  .PHONY: image
223  image:
224    [ -d $(IMAGEDIR) ] || mkdir $(IMAGEDIR)
225    $(MAKEARCH) -C $(VENDDIR) image
226
227  #
228  # fancy target that allows a vendor to have other top level
229  # make targets,  for example "make vendor_flash" will run the
230  # vendor_flash target in the vendors directory
231  #
```

```
232
233  vendor_%:
234    $(MAKEARCH) -C $(VENDDIR) $@
235
236  .PHONY: linux
237  linux linux%_only:
238    @if [ $(LINUXDIR) != linux-2.5.x -a ! -f $(LINUXDIR)/.depend ] ; then \
239        echo "ERROR: you need to do a 'make dep' first" ; \
240        exit 1 ; \
241    fi
242    $(MAKEARCH_KERNEL) -C $(LINUXDIR) $(LINUXTARGET) || exit 1
243    if [ -f $(LINUXDIR)/vmlinux ]; then \
244        ln -f $(LINUXDIR)/vmlinux $(LINUXDIR)/linux ; \
245    fi
246
247  .PHONY: subdirs
248  subdirs: linux
249    for dir in $(DIRS) ; do $(MAKEARCH_KERNEL) -C $$dir || exit 1 ; done
250
251  dep:
252    @if [ ! -f $(LINUXDIR)/.config ] ; then \
253        echo "ERROR: you need to do a 'make config' first" ; \
254        exit 1 ; \
255    fi
256    $(MAKEARCH_KERNEL) -C $(LINUXDIR) dep
257
258  # This one removes all executables from the tree and forces their relinking
259  .PHONY: relink
260  relink:
261    find user -name '*.gdb' | sed 's/^\(.*\)\.gdb/\1 \1.gdb/' | xargs rm -f
262    find $(VENDDIR) -name '*.gdb' | sed 's/^\(.*\)\.gdb/\1 \1.gdb/' | xargs rm -f
263
264  clean: modules_clean
265    for dir in $(LINUXDIR) $(DIRS); do $(MAKEARCH) -C $$dir clean ; done
266    rm -rf $(ROMFSDIR)/*
267    rm -f $(IMAGEDIR)/*
268    rm -f config.tk
269    rm -f $(LINUXDIR)/linux
270    rm -rf $(LINUXDIR)/net/ipsec/alg/libaes $(LINUXDIR)/net/ipsec/alg/perlasm
271
272  real_clean mrproper: clean
273    -$(MAKEARCH_KERNEL) -C $(LINUXDIR) mrproper
274    -$(MAKEARCH) -C config clean
275    rm -rf romfs config.in config.arch config.tk images
276    rm -f modules/config.tk
277    rm -rf .config .config.old .oldconfig autoconf.h
278
```

```
279  distclean: mrproper
280    -$(MAKEARCH_KERNEL) -C $(LINUXDIR) distclean
281
282  %_only:
283    [ ! -d "$(@:_only=)" ] || $(MAKEARCH) -C $(@:_only=)
284
285  %_clean:
286    [ ! -d "$(@:_clean=)" ] || $(MAKEARCH) -C $(@:_clean=) clean
287
288  config_error:
289    @echo "*************************************************"
290    @echo "You have not run make config."
291    @echo "The build sequence for this source tree is:"
292    @echo "1. 'make config' or 'make xconfig'"
293    @echo "2. 'make dep'"
294    @echo "3. 'make'"
295    @echo "*************************************************"
296    @exit 1
297
298  prune:
299    $(MAKE) -C user prune
300
301  dist-prep:
302    -find $(ROOTDIR) -name 'Makefile*.bin' | while read t; do \
303        $(MAKEARCH) -C `dirname $$t` -f `basename $$t` $@; \
304     done
305
306  ###############################################################################
```

uClinux-dist/linux-2.4.x/arch/armnommu/Makefile

```
1   #
2   # arch/armnommu/Makefile
3   #
4   # This file is subject to the terms and conditions of the GNU General Public
5   # License.  See the file "COPYING" in the main directory of this archive
6   # for more details.
7   #
8   # Copyright (C) 1995-2001 by Russell King
9   # Addition of S3C4530 by Arcturus Networks Inc.
10
11   LINKFLAGS  :=-p -X -T arch/armnommu/vmlinux.lds
12   GZFLAGS        :=-9
13   CFLAGS     +=-fno-common -pipe -fno-builtin -D__linux__
14
15   ifneq ($(CONFIG_NO_FRAME_POINTER),y)
16   CFLAGS     :=$(CFLAGS:-fomit-frame-pointer=)
17   endif
18
19   ifeq ($(CONFIG_DEBUG_INFO),y)
20   CFLAGS     +=-g
21   endif
22
23   CFLAGS += -DNO_MM
24   AFLAGS += -DNO_MM
25
26   # Select CPU dependent flags.  Note that order of declaration is important;
27   # the options further down the list override previous items.
28   #
29   apcs-$(CONFIG_CPU_26)      :=-mapcs-26 -mcpu=arm3 -Os
30   apcs-$(CONFIG_CPU_32)      :=-mapcs-32
31
32   arch-$(CONFIG_CPU_32v3)        :=-march=armv3
33   arch-$(CONFIG_CPU_32v4)        :=-march=armv4
34   arch-$(CONFIG_CPU_32v5)        :=-march=armv5
35
36   proc-$(CONFIG_CPU_32v3)        :=-march=armv3m
37   proc-$(CONFIG_CPU_32v4)        :=-march=armv4
38   proc-$(CONFIG_CPU_32v5)        :=-march=armv5
39
40   tune-$(CONFIG_CPU_ARM7V3)  :=-mtune=arm610
41   tune-$(CONFIG_CPU_ARM610)  :=-mtune=arm610
42   #tune-$(CONFIG_CPU_ARM710) :=-mtune=arm710
43   tune-$(CONFIG_CPU_ARM710)  :=-mtune=arm7tdmi
44   tune-$(CONFIG_CPU_ARM720T) :=-mtune=arm7tdmi
45   tune-$(CONFIG_CPU_ARM740T) :=-mtune=arm7tdmi
```

```
46  tune-$(CONFIG_CPU_ARM920T) :=-mtune=arm9tdmi
47  tune-$(CONFIG_CPU_SA110)   :=-mtune=strongarm110
48  tune-$(CONFIG_CPU_SA1100)  :=-mtune=strongarm1100
49
50  ifeq ($(CONFIG_CPU_BIG_ENDIAN),y)
51  CFLAGS     += -mbig-endian
52  AFLAGS     += -mbig-endian
53  LINKFLAGS  += -EB
54  LDFLAGS        += -EB
55  endif
56
57  CFLAGS     += $(apcs-y) $(arch-y) $(tune-y) -mshort-load-bytes -msoft-float
58  AFLAGS     += $(apcs-y) $(proc-y) -msoft-float -mno-fpu
59
60  LIBGCC     := $(shell $(CC) $(CFLAGS) --print-libgcc-file-name)
61
62  ifeq ($(CONFIG_CPU_26),y)
63  PROCESSOR   = armo
64  TEXTADDR    = 0x02080000
65  endif
66
67  ifeq ($(CONFIG_CPU_32),y)
68  PROCESSOR   = armv
69  TEXTADDR    = 0xC0008000
70  endif
71
72  ifeq ($(CONFIG_ARCH_ARCA5K),y)
73  MACHINE         = arc
74  endif
75
76  ifeq ($(CONFIG_ARCH_RPC),y)
77  MACHINE         = rpc
78  endif
79
80  ifeq ($(CONFIG_ARCH_EBSA110),y)
81  MACHINE         = ebsa110
82  endif
83
84  ifeq ($(CONFIG_ARCH_CLPS7500),y)
85  MACHINE         = clps7500
86  INCDIR      = cl7500
87  endif
88
89  ifeq ($(CONFIG_FOOTBRIDGE),y)
90  MACHINE         = footbridge
91  INCDIR      = ebsa285
92  endif
```

```
93
94   ifeq ($(CONFIG_ARCH_CO285),y)
95   TEXTADDR    = 0x60008000
96   MACHINE         = footbridge
97   INCDIR      = ebsa285
98   endif
99
100   ifeq ($(CONFIG_ARCH_NEXUSPCI),y)
101   MACHINE        = nexuspci
102   endif
103
104   ifeq ($(CONFIG_ARCH_SHARK),y)
105   MACHINE        = shark
106   endif
107
108   ifeq ($(CONFIG_ARCH_SA1100),y)
109   ifeq ($(CONFIG_SA1111),y)
110   # SA1111 DMA bug: we don't want the kernel to live in precious DMA-able memory
111   TEXTADDR   = 0xc0208000
112   endif
113   MACHINE        = sa1100
114   endif
115
116   ifeq ($(CONFIG_ARCH_L7200),y)
117   MACHINE        = l7200
118   endif
119
120   ifeq ($(CONFIG_ARCH_INTEGRATOR),y)
121   MACHINE        = integrator
122   TEXTADDR   = 0x00008000
123   endif
124
125   ifeq ($(CONFIG_ARCH_CLPS711X),y)
126   TEXTADDR   = 0xc0018000
127   MACHINE        = clps711x
128   endif
129
130   ifeq ($(CONFIG_ARCH_DSC21),y)
131   MACHINE    = dsc21
132   TEXTADDR   = 0x02000000
133   DATAADDR   = 0x08000000
134   INCDIR     = dsc21
135   #LINKFLAGS     = -X -T arch/armnommu/vmlinux.lds
136   endif
137
138   ifeq ($(CONFIG_ARCH_P52),y)
139   TEXTADDR   = 0x900000
```

```
140  MACHINE        = p52
141  CFLAGS        += -DNO_MM -DMAGIC_ROM_PTR
142  endif
143
144  ifeq ($(CONFIG_ARCH_SPIPE),y)
145  TEXTADDR   = 0x808000
146  MACHINE        = spipe
147  endif
148
149  ifeq ($(CONFIG_ARCH_ATMEL),y)
150  TEXTADDR  = 0x1000000
151  MACHINE       = atmel
152  endif
153
154  ifeq ($(CONFIG_ARCH_NETARM),y)
155  MACHINE    = netarm
156  TEXTADDR   = 0x00008000
157  INCDIR     = netarm
158  CFLAGS        += -mcpu=arm7tdmi
159  #LINKFLAGS     = -X -T arch/armnommu/vmlinux.lds
160  endif
161
162  ifeq ($(CONFIG_ARCH_SWARM),y)
163  MACHINE    = swarm
164  TEXTADDR   = 0x00008000
165  endif
166
167  ifeq ($(CONFIG_BOARD_SNDS100),y)
168  TEXTADDR       = 0x00008000
169  MACHINE        = snds100
170  endif
171
172  ifeq ($(CONFIG_ARCH_S3C44B0), y)
173  TEXTADDR   = 0x0C008000
174  MACHINE        = s3c44b0
175  endif
176  ifeq ($(CONFIG_BOARD_EVS3C4530HEI),y)
177  TEXTADDR         = 0x00020000
178  MACHINE          = evS3C4530HEI
179  INCDIR           = $(MACHINE)
180  endif
181
182  ifeq ($(CONFIG_BOARD_SMDK40100),y)
183  TEXTADDR       = 0x00080000
184  MACHINE          = S3C3410
185  INCDIR           = $(MACHINE)
186  endif
```

```
187
188   export    LDFLAGS LIBGCC MACHINE PROCESSOR TEXTADDR GZFLAGS
189
190   # Only set INCDIR if its not already defined above
191   # Grr, ?= doesn't work as all the other assignment operators do.  Make bug?
192   ifeq ($(origin INCDIR), undefined)
193   INCDIR        := $(MACHINE)
194   endif
195
196   # If we have a machine-specific directory, then include it in the build.
197   MACHDIR       := arch/armnommu/mach-$(MACHINE)
198   ifeq ($(MACHDIR),$(wildcard $(MACHDIR)))
199   SUBDIRS       += $(MACHDIR)
200   CORE_FILES    := $(MACHDIR)/$(MACHINE).o $(CORE_FILES)
201   endif
202
203   ifeq ($(CONFIG_ARCH_DSC21),y)
204   HEAD      := arch/armnommu/kernel/head-arm-dsc21.o \
205             arch/armnommu/kernel/init_task.o
206   else
207   HEAD      := arch/armnommu/kernel/head-$(PROCESSOR).o \
208             arch/armnommu/kernel/init_task.o
209   endif
210   SUBDIRS       +=  arch/armnommu/kernel  arch/armnommu/mm  arch/armnommu/lib
arch/armnommu/nwfpe
211   CORE_FILES    :=     arch/armnommu/kernel/kernel.o     arch/armnommu/mm/mm.o
$(CORE_FILES)
212   LIBS      := arch/armnommu/lib/lib.a $(LIBS) $(LIBGCC)
213
214   ifeq ($(CONFIG_NWFPE),y)
215   LIBS      := arch/armnommu/nwfpe/math-emu.o $(LIBS)
216   endif
217
218   ifeq ($(CONFIG_ARCH_CLPS7500),y)
219   SUBDIRS       += drivers/acorn/char
220   DRIVERS       += drivers/acorn/char/acorn-char.o
221   endif
222
223   ifeq ($(CONFIG_ARCH_NETARM),y)
224   HEAD      := arch/armnommu/kernel/head-arm-netarm.o \
225             arch/armnommu/kernel/init_task.o
226   endif
227
228   MAKEBOOT   = $(MAKE) -C arch/$(ARCH)/boot LINUX=$(LINUX)
229   MAKETOOLS  = $(MAKE) -C arch/$(ARCH)/tools LINUX=$(LINUX)
230
231   # The following is a hack to get 'constants.h' up
```

```
232   # to date before starting compilation
233
234   $(patsubst %,_dir_%, $(SUBDIRS)): maketools
235   $(patsubst %,_modsubdir_%,$(MOD_DIRS)): maketools
236
237   symlinks: archsymlinks
238
239   archsymlinks:
240     $(RM) include/asm-armnommu/arch include/asm-armnommu/proc
241     (cd include/asm-armnommu; ln -sf arch-$(INCDIR) arch; ln -sf proc-$(PROCESSOR)
proc)
242
243   $(LINUX): arch/armnommu/vmlinux.lds
244
245   arch/armnommu/vmlinux.lds: arch/armnommu/vmlinux-$(PROCESSOR).lds.in dummy
246   ifeq ($(CONFIG_ARCH_DSC21),y)
247     @sed 's/TEXTADDR/$(TEXTADDR)/' <$< >tmp.ld
248     @sed 's/DATAADDR/$(DATAADDR)/' <tmp.ld >$@
249     $(RM) tmp.ld
250   else
251     @sed 's/TEXTADDR/$(TEXTADDR)/' <$< >$@
252   endif
253
254   arch/armnommu/kernel arch/armnommu/mm arch/armnommu/lib: dummy
255     $(MAKE) CFLAGS="$(CFLAGS) $(CFLAGS_KERNEL)" $(subst $@, _dir_$@, $@)
256
257   bzImage zImage zinstall Image bootpImage install: $(LINUX)
258     @$(MAKEBOOT) $@
259
260   CLEAN_FILES   += \
261     arch/armnommu/vmlinux.lds \
262     include/asm-armnommu/constants.h* \
263     include/asm-armnommu/mach-types.h
264
265   MRPROPER_FILES    += \
266     include/asm-armnommu/arch \
267     include/asm-armnommu/proc
268
269   # We use MRPROPER_FILES and CLEAN_FILES now
270   archmrproper:
271     @/bin/true
272
273   archclean:
274     @$(MAKEBOOT) clean
275
276   archdep: scripts/mkdep archsymlinks
277     @$(MAKETOOLS) dep
```

```
278     @$(MAKEBOOT) dep
279
280   maketools: checkbin
281     @$(MAKETOOLS) all
282
283   # Ensure this is ld "2.9.4" or later
284   NEW_LINKER    := $(shell $(LD) --gc-sections --version >/dev/null 2>&1; echo
$$?)
285
286   ifneq ($(NEW_LINKER),0)
287   checkbin:
288     @echo '*** ${VERSION}.${PATCHLEVEL} kernels no longer build correctly with old
versions of binutils.'
289     @echo '*** Please upgrade your binutils to 2.9.5.'
290     @false
291   else
292   checkbin:
293     @true
294   endif
295
296   # My testing targets (that short circuit a few dependencies)
297   zImg:;    @$(MAKEBOOT) zImage
298   Img:; @$(MAKEBOOT) Image
299   i:;   @$(MAKEBOOT) install
300   zi:;  @$(MAKEBOOT) zinstall
301   bp:;  @$(MAKEBOOT) bootpImage
302
303   #
304   # Configuration targets.  Use these to select a
305   # configuration for your architecture
306   %_config:
307     @( \
308     CFG=$(@:_config=); \
309     if [ -f arch/armnommu/def-configs/$$CFG ]; then \
310       $(RM) arch/armnommu/defconfig; \
311       cp arch/armnommu/def-configs/$$CFG arch/armnommu/defconfig; \
312       echo "*** Default configuration for $$CFG installed"; \
313       echo "*** Next, you may run 'make oldconfig'"; \
314     else \
315       echo "$$CFG does not exist"; \
316     fi; \
317     )
```

uClinux-dist/linux-2.4.x/arch/armnommu/config.in

```
#
# For a description of the syntax of this configuration file,
# see Documentation/kbuild/config-language.txt.
#
mainmenu_name "Linux Kernel Configuration"

define_bool CONFIG_ARM y
define_bool CONFIG_SBUS n
define_bool CONFIG_UID16 y
 define_bool CONFIG_RWSEM_GENERIC_SPINLOCK y

 # Begin uclinux additions ----------------------------------------------------
 define_bool CONFIG_UCLINUX y
 define_bool MAGIC_ROM_PTR y
 # End uclinux additions ------------------------------------------------------


#-----------------------------------------------------------------------------
#                         C o d e   m a t u r i t y
#-----------------------------------------------------------------------------
 mainmenu_option next_comment
 comment 'Code maturity level options'
 bool 'Prompt for development and/or incomplete code/drivers' CONFIG_EXPERIMENTAL
 bool 'Prompt for obsolete code/drivers' CONFIG_OBSOLETE
 endmenu

#-----------------------------------------------------------------------------
#                       L o a d a b l e   M o d u l e
#-----------------------------------------------------------------------------
 mainmenu_option next_comment
 comment 'Loadable module support'
 bool 'Enable loadable module support' CONFIG_MODULES
 if [ "$CONFIG_MODULES" = "y" ]; then
    bool '  Set version information on all module symbols' CONFIG_MODVERSIONS
    bool '  Kernel module loader' CONFIG_KMOD
 fi
 endmenu

#-----------------------------------------------------------------------------
#                              S y s t e m
#-----------------------------------------------------------------------------
```

```
40
41  mainmenu_option next_comment
42  comment 'System Type'
43  choice 'ARM system type'    \
44      "TI-DSC21      CONFIG_ARCH_DSC21 \
45       Conexant      CONFIG_ARCH_CNXT \
46       NET+ARM       CONFIG_ARCH_NETARM \
47       SWARM         CONFIG_ARCH_SWARM \
48       Samsung       CONFIG_ARCH_SAMSUNG \
49       Atmel         CONFIG_ARCH_ATMEL" TI-DSC21
50
51  bool 'Generate big endian code' CONFIG_CPU_BIG_ENDIAN
52
53  if [ "$CONFIG_ARCH_CNXT" = "y" ]; then
54  choice ' Conexant/Mindspeed architecture' \
55         " P52xxCtrl     CONFIG_ARCH_P52 \
56        sp_CN9414         CONFIG_ARCH_SPIPE" P52xxCtrl
57
58    if [ "$CONFIG_ARCH_P52" = "y" ]; then
59    choice ' P52xx board implementation' \
60     "IAD_EVM        CONFIG_IAD_EVM \
61      JSCHornet      CONFIG_HORNET" IAD_EVM
62    fi
63  fi
64  if [ "$CONFIG_ARCH_NETARM" = "y" ]; then
65  choice ' NET+ARM Processor type' \
66         " NET+15        CONFIG_NETARM_NET15 \
67           NET+40        CONFIG_NETARM_NET40 \
68           NET+50        CONFIG_NETARM_NET50" NET+40
69  fi
70
71  bool 'Set flash/sdram size and base addr' CONFIG_SET_MEM_PARAM
72  if [ "$CONFIG_SET_MEM_PARAM" = "y" ]; then
73     hex '(S)DRAM Base Address' DRAM_BASE 0x00800000
74     hex '(S)DRAM Size ' DRAM_SIZE 0x00800000
75     hex 'FLASH Base Address ' FLASH_MEM_BASE 0x00400000
76     hex 'FLASH Size ' FLASH_SIZE 0x00400000
77  fi
78
79  choice 'Kernel executes from' \
80     "RAM    CONFIG_RAMKERNEL \
81      ROM    CONFIG_ROMKERNEL" ROM
82
83  # ARM940T
84  if [ "$CONFIG_ARCH_CNXT" = "y" ]; then
85     define_bool CONFIG_CPU_32 y
86     define_bool CONFIG_CPU_26 n
```

```
87      define_bool CONFIG_CPU_ARM940T y
88      define_bool CONFIG_NO_PGT_CACHE y
89      define_bool CONFIG_CPU_WITH_CACHE y
90      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION y
91   if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
92      define_hex DRAM_BASE 0x00800000
93      define_hex DRAM_SIZE 0x00800000
94      define_hex FLASH_MEM_BASE 0x00400000
95      define_hex FLASH_SIZE 0x00400000
96   fi
97
98      bool '  ARM940T CPU idle' CONFIG_CPU_ARM940_CPU_IDLE
99      bool '  ARM940T I-Cache on' CONFIG_CPU_ARM940_I_CACHE_ON
100      bool '  ARM940T D-Cache on' CONFIG_CPU_ARM940_D_CACHE_ON
101   if [ "$CONFIG_CPU_ARM940_D_CACHE_ON" = "y" ] ; then
102                bool '      Force   write   through   caches   on   ARM940T'
CONFIG_CPU_ARM940_WRITETHROUGH
103   fi
104   fi
105
106
107   if [ "$CONFIG_ARCH_DSC21" = "y" ]; then
108      define_bool CONFIG_CPU_ARM710 y
109      define_bool CONFIG_CPU_32 y
110      define_bool CONFIG_CPU_26 n
111      define_bool CONFIG_NO_PGT_CACHE y
112      define_bool CONFIG_CPU_WITH_CACHE y
113      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION y
114      if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
115      define_hex DRAM_BASE 0x08000000
116      define_hex DRAM_SIZE 0x00200000
117      define_hex FLASH_MEM_BASE 0x08400000
118      define_hex FLASH_SIZE 0x00200000
119      fi
120      define_bool CONFIG_DUMMY_CONSOLE y
121   fi
122
123   if [ "$CONFIG_ARCH_SWARM" = "y" ]; then
124   #  define_bool CONFIG_CPU_ARM610 y
125      define_bool CONFIG_CPU_32 y
126      define_bool CONFIG_CPU_26 n
127      define_bool CONFIG_CPU_32v3 y
128      define_bool CONFIG_CPU_ARM7V3 y
129      define_bool CONFIG_NO_PGT_CACHE y
130      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION y
131      define_hex FLASH_MEM_BASE 0x00000000
132      define_hex FLASH_SIZE 0x00100000
```

```
133   fi
134
135   if [ "$CONFIG_ARCH_ATMEL" = "y" ]; then
136      define_bool CONFIG_NO_PGT_CACHE y
137      define_bool CONFIG_CPU_ARM710 y
138      define_bool CONFIG_CPU_32 y
139      define_bool CONFIG_CPU_32v4 y
140      define_bool CONFIG_CPU_WITH_CACHE n
141      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
142      define_bool CONFIG_SERIAL_ATMEL y
143
144      bool 'Serial Console' CONFIG_SERIAL_ATMEL_CONSOLE
145
146      if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
147         define_hex DRAM_BASE 0x01000000
148           define_hex DRAM_SIZE 0x00600000
149           define_hex FLASH_MEM_BASE 0x01600000
150         define_hex FLASH_SIZE 0x00200000
151      fi
152
153       hex 'Memory mapped 16-bit io base' CONFIG_MEM16_BASE 0x03000000
154       hex 'Memory mapped 8-bit io base' CONFIG_MEM8_BASE 0x03000000
155       hex '16-bit io base' CONFIG_IO16_BASE 0x02000000
156       hex '8-bit io base' CONFIG_IO8_BASE 0x02000000
157
158      choice ' Atmel CPU' \
159         " AT91x40       CONFIG_CPU_AT91X40 \
160        AT91x63       CONFIG_CPU_AT91X63" AT91x40
161
162      bool 'Atmel Kernel-Debug hack' CONFIG_ATMEL_DEBUG
163      if [ "$CONFIG_ATMEL_DEBUG" = "y" ]; then
164     hex 'Debug buffer address' AT91_DEBUG_BASE 0x01400000
165      fi
166   fi
167
168   if [ "$CONFIG_ARCH_NETARM" = "y" ]; then
169      define_bool CONFIG_CPU_ARM710 y
170      define_bool CONFIG_CPU_ARM7TDMI y
171      define_bool CONFIG_CPU_32 y
172      define_bool CONFIG_CPU_26 n
173      define_bool CONFIG_CPU_WITH_CACHE y
174      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
175      define_bool CONFIG_NO_PGT_CACHE y
176      if [ "$CONFIG_NETARM_NET40" = "y" ]; then
177         bool 'NET+ARM NET+40 Rev2' CONFIG_NETARM_NET40_REV2
178         if [ "$CONFIG_NETARM_NET40_REV2" = "n" ]; then
179            bool 'NET+ARM NET+40 Rev4' CONFIG_NETARM_NET40_REV4
```

```
180         fi
181         bool 'NET+ARM PLL Bypass Patch' CONFIG_NETARM_PLL_BYPASS
182         bool 'NET+ARM EMLIN Board' CONFIG_NETARM_EMLIN
183      fi
184   #  default memory configuration for Net40 and EMLIN boards
185      if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
186         define_hex DRAM_BASE 0x00000000
187         define_hex FLASH_MEM_BASE 0x10000000
188         if [ "$CONFIG_NETARM_EMLIN" = "y" ]; then
189            define_hex DRAM_SIZE 0x01000000
190            define_hex FLASH_SIZE 0x00200000
191         fi
192         if [ "$CONFIG_NETARM_EMLIN" = "n" ]; then
193            define_hex DRAM_SIZE 0x02000000
194            define_hex FLASH_SIZE 0x00800000
195      define_bool CONFIG_NETARM_EEPROM y
196         fi
197      fi
198   fi
199
200   if [ "$CONFIG_ARCH_SAMSUNG" = "y" ]; then
201      choice 'Board Implementation' \
202      "S3C3410-SMDK40100 CONFIG_BOARD_SMDK40100 \
203      S3C44B0X-51EDA CONFIG_ARCH_S3C44B0 \
204      S3C4530-HEI    CONFIG_BOARD_EVS3C4530HEI \
205      S3C4510-SNDS100    CONFIG_BOARD_SNDS100" S3C4510-SNDS100
206   fi
207
208   if [ "$CONFIG_ARCH_S3C44B0" = "y" ]; then
209      define_bool CONFIG_NO_PGT_CACHE y
210      define_bool CONFIG_CPU_32 y
211      define_bool CONFIG_CPU_26 n
212      define_bool CONFIG_CPU_ARM710 y
213      define_bool CONFIG_CPU_WITH_CACHE y
214      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
215      define_bool CONFIG_SERIAL_S3C44B0 y
216      if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
217         define_hex DRAM_BASE 0x0c000000
218         define_hex DRAM_SIZE 0x01000000
219         define_hex FLASH_MEM_BASE 0x00000000
220         define_hex FLASH_SIZE 0x00200000
221      fi
222   fi
223   if [ "$CONFIG_BOARD_SMDK40100" = "y" ]; then
224       define_string CONFIG_SPU_NAME "S3C3410X"
225       define_bool CONFIG_CPU_S3C3410      y
226       define_bool CONFIG_CPU_ARM710       y
```

```
227       define_bool CONFIG_CPU_32v4         y
228       define_bool CONFIG_CPU_32           y
229       define_bool CONFIG_CPU_26           n
230       define_bool CONFIG_NO_PGT_CACHE     y
231       define_bool CONFIG_CPU_WITH_CACHE   y
232       define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
233       define_int  CONFIG_ARM_CLK          40000000
234       define_bool CONFIG_SERIAL_S3C3410   y
235       if [ "$CONFIG_SET_MEM_PARAM" = "n" ]; then
236          define_hex DRAM_BASE 0x00000000
237          define_hex DRAM_SIZE 0x00800000
238          define_hex FLASH_MEM_BASE 0x01000000
239          define_hex FLASH_SIZE 0x00200000
240       fi
241    fi
242
243    if [ "$CONFIG_BOARD_EVS3C4530HEI" = "y" ]; then
244       define_string CONFIG_CPU_NAME "S3C4530A01-Q"
245       bool ' Enable uCbootloader support' CONFIG_UCBOOTSTRAP
246       define_bool CONFIG_CPU_S3C4530      y
247       define_bool CONFIG_CPU_ARM710       y
248       define_bool CONFIG_CPU_32           y
249       define_bool CONFIG_CPU_26           n
250       define_bool CONFIG_NO_PGT_CACHE     y
251       define_bool CONFIG_CPU_WITH_CACHE   y
252       define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
253       define_int  CONFIG_ARM_CLK          50000000
254       define_hex  DRAM_BASE               0x00000000
255       define_hex  DRAM_SIZE               0x007F0000
256       define_hex  FLASH_MEM_BASE          0x01000000
257       define_hex  FLASH_SIZE              0x00200000
258       define_hex  FLASH1_MEM_BASE         0x01200000
259       define_hex  FLASH1_SIZE             0x00200000
260    fi
261
262    if [ "$CONFIG_BOARD_SNDS100" = "y" ]; then
263       define_bool CONFIG_NO_PGT_CACHE y
264       define_bool CONFIG_CPU_32 y
265       define_bool CONFIG_CPU_26 n
266       define_bool CONFIG_CPU_ARM710 y
267       define_bool CONFIG_CPU_WITH_CACHE y
268       define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION n
269       define_bool CONFIG_SERIAL_SAMSUNG y
270       define_hex DRAM_BASE 0x00000000
271       define_hex DRAM_SIZE 0x00800000
272       define_hex FLASH_MEM_BASE 0x01000000
273       define_hex FLASH_SIZE 0x00200000
```

```
274   fi
275
276   if [ "$CONFIG_ARCH_INTEGRATOR" = "y" ]; then
277      define_bool CONFIG_NO_PGT_CACHE y
278      define_bool CONFIG_CPU_ARM740T y
279      define_bool CONFIG_CPU_32 y
280      define_bool CONFIG_CPU_WITH_CACHE y
281      define_bool CONFIG_CPU_WITH_MCR_INSTRUCTION y
282      define_hex DRAM_BASE 0
283      define_hex DRAM_SIZE 0x04000000
284      define_hex FLASH_MEM_BASE 0x25000000
285      define_hex FLASH_SIZE 0x02000000
286   fi
287   endmenu
288
289
#------------------------------------------------------------------------------
290   #                               G e n e r a l
291
#------------------------------------------------------------------------------
292   mainmenu_option next_comment
293   comment 'General setup'
294   source drivers/pci/Config.in
295   bool 'Support hot-pluggable devices' CONFIG_HOTPLUG
296   if [ "$CONFIG_HOTPLUG" = "y" ]; then
297      source drivers/pcmcia/Config.in
298   else
299      define_bool CONFIG_PCMCIA n
300   fi
301   bool 'Networking support' CONFIG_NET
302   bool 'System V IPC' CONFIG_SYSVIPC
303   bool 'Reduced memory footprint' CONFIG_REDUCED_MEMORY
304   bool 'BSD Process Accounting' CONFIG_BSD_PROCESS_ACCT
305   bool 'Sysctl support' CONFIG_SYSCTL
306   tristate 'NWFPE math emulation' CONFIG_NWFPE
307   choice 'Kernel core (/proc/kcore) format' \
308     "ELF        CONFIG_KCORE_ELF    \
309      A.OUT      CONFIG_KCORE_AOUT" ELF
310   define_bool CONFIG_BINFMT_FLAT y
311   define_bool CONFIG_KERNEL_ELF y
312
313   if [ "$CONFIG_EXPERIMENTAL" = "y" ]; then
314      bool 'Power Management support' CONFIG_PM
315   fi
316
317   if [ "$CONFIG_CPU_32" = "y" ]; then
318      tristate 'RISC OS personality' CONFIG_ARTHUR
```

```
319   fi
320   if [ "$CONFIG_ARCH_EBSA110" = "y" -o \
321        "$CONFIG_ARCH_SA1100" = "y" -o \
322        "$CONFIG_ARCH_CLPS7500" = "y" -o \
323        "$CONFIG_ARCH_PERSONAL_SERVER" = "y" -o \
324        "$CONFIG_ARCH_CATS" = "y" -o \
325        "$CONFIG_ARCH_ATMEL" = "y" ]; then
326      bool 'Compiled-in Kernel Boot Parameter' CONFIG_CMDLINE_BOOL
327      if [ "$CONFIG_CMDLINE_BOOL" = "y" ]; then
328         string 'Default kernel command string' CONFIG_CMDLINE ""
329      fi
330   fi
331   if [ "$CONFIG_ARCH_NETWINDER" = "y" -o \
332        "$CONFIG_ARCH_EBSA110" = "y" -o   \
333        "$CONFIG_ARCH_EBSA285" = "y" -o   \
334        "$CONFIG_ARCH_CO285" = "y" -o     \
335        "$CONFIG_ARCH_SA1100" = "y" ]; then
336      bool 'Timer and CPU usage LEDs' CONFIG_LEDS
337      if [ "$CONFIG_LEDS" = "y" ]; then
338         if [ "$CONFIG_ARCH_NETWINDER" = "y" -o \
339        "$CONFIG_ARCH_EBSA285" = "y" -o   \
340        "$CONFIG_ARCH_CO285" = "y" -o     \
341        "$CONFIG_ARCH_SA1100" = "y" ]; then
342            bool '  Timer LED' CONFIG_LEDS_TIMER
343            bool '  CPU usage LED' CONFIG_LEDS_CPU
344         fi
345      fi
346      if [ "$CONFIG_ARCH_EBSA110" = "y" ]; then
347         define_bool CONFIG_LEDS_TIMER y
348      fi
349   fi
350   if [ "$CONFIG_CPU_32" = "y" -a "$CONFIG_ARCH_EBSA110" != "y" ]; then
351      bool 'Kernel-mode alignment trap handler' CONFIG_ALIGNMENT_TRAP
352   fi
353
354   if [ "$CONFIG_ARCH_INTEGRATOR" = "y" ]; then
355      bool 'PCI support' CONFIG_PCI_INTEGRATOR
356      define_bool CONFIG_PCI $CONFIG_PCI_INTEGRATOR
357   fi
358
359   endmenu
360
#------------------------------------------------------------------------------
361   #                           N e t    D e v i c e s
362
#------------------------------------------------------------------------------
363   if [ "$CONFIG_NET" = "y" ]; then
```

```
364       source net/Config.in
365       mainmenu_option next_comment
366       comment 'Network device support'
367       bool 'Network device support?' CONFIG_NETDEVICES
368       if [ "$CONFIG_NETDEVICES" = "y" ]; then
369          source drivers/net/Config.in
370       fi
371       endmenu
372       source net/ax25/Config.in
373       source net/irda/Config.in
374    fi
375
376    #-----------------------------------------------------------------------
377    #                                A T E / I D E
378    #-----------------------------------------------------------------------
379    mainmenu_option next_comment
380    comment 'ATA/IDE/MFM/RLL support'
381    tristate 'ATA/IDE/MFM/RLL support' CONFIG_IDE
382    if [ "$CONFIG_IDE" != "n" ]; then
383      source drivers/ide/Config.in
384    else
385      define_bool CONFIG_BLK_DEV_IDE_MODES n
386      define_bool CONFIG_BLK_DEV_HD n
387    fi
388    endmenu
389
390    #-----------------------------------------------------------------------
391    #                                  S C S I
392    #-----------------------------------------------------------------------
393    mainmenu_option next_comment
394    comment 'SCSI support'
395    tristate 'SCSI support?' CONFIG_SCSI
396    if [ "$CONFIG_SCSI" != "n" ]; then
397       source drivers/scsi/Config.in
398    fi
399    endmenu
400
401    #-----------------------------------------------------------------------
402    #                                  I S D N
403    #-----------------------------------------------------------------------
404    mainmenu_option next_comment
405    comment 'ISDN subsystem'
406    tristate 'ISDN support' CONFIG_ISDN
407    if [ "$CONFIG_ISDN" != "n" ]; then
```

```
405      source drivers/isdn/Config.in
406   fi
407   endmenu
408
409   #-----------------------------------------------------------------------
410   #                              C o n s o l e
411   #-----------------------------------------------------------------------
412   if [ "$CONFIG_VT" = "y" ]; then
413      mainmenu_option next_comment
414      comment 'Console drivers'
415      if [ "$CONFIG_ARCH_ACORN" != "y" -a "$CONFIG_ARCH_EBSA110" != "y" ]; then
416         bool 'VGA text console' CONFIG_VGA_CONSOLE
417      fi
418      bool 'Support Frame buffer devices' CONFIG_FB
419      source drivers/video/Config.in
420      endmenu
421   fi
422
423   #-----------------------------------------------------------------------
424   #                         M i s c    D r i v e r s
425   #-----------------------------------------------------------------------
426   source drivers/parport/Config.in
427   source drivers/mtd/Config.in
428   source drivers/pnp/Config.in
429   source drivers/block/Config.in
430   source fs/Config.in
431   source drivers/char/Config.in
432   source drivers/usb/Config.in
433   source drivers/ieee1394/Config.in
434   source drivers/message/i2o/Config.in
435   #-----------------------------------------------------------------------
436   #                      K e r n e l    H a c k i n g
437   #-----------------------------------------------------------------------
438   mainmenu_option next_comment
439   comment 'Kernel hacking'
440
441   define_bool CONFIG_FRAME_POINTER y
442   bool 'Find REVISITS' CONFIG_REVISIT
443   bool 'Verbose kernel error messages' CONFIG_DEBUG_ERRORS
444   bool 'Verbose user fault messages' CONFIG_DEBUG_USER
445   bool 'Include debugging information in kernel binary' CONFIG_DEBUG_INFO
446   dep_bool 'Magic SysRq key' CONFIG_MAGIC_SYSRQ $CONFIG_VT
```

```
446   if [ "$CONFIG_CPU_26" = "y" ]; then
447      bool 'Disable pgtable cache' CONFIG_NO_PGT_CACHE
448   fi
449   if [ "$CONFIG_EXPERIMENTAL" = "y" ]; then
450      bool 'Kernel low-level debugging functions' CONFIG_DEBUG_LL
451      if [ "$CONFIG_DEBUG_LL" = "y" ]; then
452         if [ "$CONFIG_FOOTBRIDGE" = "y" ]; then
453            bool 'Kernel low-level debugging messages via footbridge serial port'
CONFIG_DEBUG_DC21285_PORT
454         fi
455      fi
456                 bool   'Non   power-of-2   kernel   allocator   (EXPERIMENTAL)'
CONFIG_CONTIGUOUS_PAGE_ALLOC
457                    dep_bool   '       include   /proc/mem_map'   CONFIG_MEM_MAP
$CONFIG_CONTIGUOUS_PAGE_ALLOC
458   fi
459   endmenu
460
461   source lib/Config.in
```

uClinux-dist/linux-2.4.x/arch/armnommu/vmlinux-armv.lds.in

```
1   /* ld script to make ARM Linux kernel
2    * taken from the i386 version by Russell King
3    * Written by Martin Mares <mj@atrey.karlin.mff.cuni.cz>
4    */
5   OUTPUT_ARCH(arm)
6   ENTRY(stext)
7   SECTIONS
8   {
9       . = TEXTADDR;
10      .init : {           /* Init code and data       */
11          _stext = .;
12          __init_begin = .;
13              *(.text.init)
14          __proc_info_begin = .;
15              *(.proc.info)
16          __proc_info_end = .;
17          __arch_info_begin = .;
18              *(.arch.info)
19          __arch_info_end = .;
20              *(.data.init)
21          . = ALIGN(16);
22          __setup_start = .;
23              *(.setup.init)
24          __setup_end = .;
25          __initcall_start = .;
26              *(.initcall.init)
27          __initcall_end = .;
28          . = ALIGN(4096);
29          __init_end = .;
30      }
31
32      /DISCARD/ : {           /* Exit code and data       */
33          *(.text.exit)
34          *(.data.exit)
35          *(.exitcall.exit)
36      }
37
38      .text : {           /* Real text segment        */
39          _text = .;      /* Text and read-only data  */
40              *(.text)
41              *(.fixup)
42              *(.gnu.warning)
43              *(.text.lock)   /* out-of-line lock text */
44              *(.rodata)
45              *(.glue_7)
```

```
46              *(.glue_7t)
47              *(.kstrtab)
48          . = ALIGN(16);
49          __start___ex_table = .; /* Exception table      */
50              *(__ex_table)
51          __stop___ex_table = .;
52
53          __start___ksymtab = .;  /* Kernel symbol table      */
54              *(__ksymtab)
55          __stop___ksymtab = .;
56
57          __start___kallsyms = .; /* All kernel symbols           */
58              *(__kallsyms)
59          __stop___kallsyms = .;
60
61          *(.got)          /* Global offset table      */
62          romfs_data = .;
63          romfs.o
64          romfs_data_end = .;
65
66          _etext = .;      /* End of text section      */
67      }
68
69      . = ALIGN(8192);
70
71      .data : {
72          /*
73           * first, the init task union, aligned
74           * to an 8192 byte boundary.
75           */
76          *(.init.task)
77
78          /*
79           * then the cacheline aligned data
80           */
81          . = ALIGN(32);
82          *(.data.cacheline_aligned)
83
84          /*
85           * and the usual data section
86           */
87          *(.data)
88          CONSTRUCTORS
89
90          _edata = .;
91      }
92
```

```
93      .bss : {
94          __bss_start = .;     /* BSS              */
95          *(.bss)
96          *(COMMON)
97          _end = . ;
98      }
99
100     . = ALIGN(8192);
101     _end_kernel = . ;
102
103                     /* Stabs debugging sections.    */
104     .stab 0 : { *(.stab) }
105     .stabstr 0 : { *(.stabstr) }
106     .stab.excl 0 : { *(.stab.excl) }
107     .stab.exclstr 0 : { *(.stab.exclstr) }
108     .stab.index 0 : { *(.stab.index) }
109     .stab.indexstr 0 : { *(.stab.indexstr) }
110     .comment 0 : { *(.comment) }
111     .debug_abbrev 0 : { *(.debug_abbrev) }
112     .debug_info 0 : { *(.debug_info) }
113     .debug_line 0 : { *(.debug_line) }
114     .debug_pubnames 0 : { *(.debug_pubnames) }
115     .debug_aranges 0 : { *(.debug_aranges) }
116 }
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/Makefile

```
1   #
2   # arch/arm/boot/Makefile
3   #
4   # This file is subject to the terms and conditions of the GNU General Public
5   # License.  See the file "COPYING" in the main directory of this archive
6   # for more details.
7   #
8   # Copyright (C) 1995-2000 Russell King
9   #
10
11   SYSTEM =$(TOPDIR)/$(LINUX)
12
13   ifeq ($(CONFIG_CPU_26),y)
14   ZTEXTADDR   = 0x02080000
15   PARAMS_PHYS     = 0x0207c000
16   INITRD_PHYS     = 0x02180000
17   INITRD_VIRT     = 0x02180000
18   endif
19
20   ifeq ($(CONFIG_ARCH_RPC),y)
21   ZTEXTADDR   = 0x10008000
22   PARAMS_PHYS     = 0x10000100
23   INITRD_PHYS     = 0x18000000
24   INITRD_VIRT     = 0xc8000000
25   endif
26
27   ifeq ($(CONFIG_ARCH_CLPS7500),y)
28   ZTEXTADDR   = 0x10008000
29   endif
30
31   ifeq ($(CONFIG_ARCH_EBSA110),y)
32   ZTEXTADDR   = 0x00008000
33   PARAMS_PHYS     = 0x00000400
34   INITRD_PHYS     = 0x00800000
35   INITRD_VIRT     = 0xc0800000
36   endif
37
38   ifeq ($(CONFIG_ARCH_SHARK),y)
39   ZTEXTADDR   = 0x08508000
40   ZRELADDR    = 0x08008000
41   endif
42
43   ifeq ($(CONFIG_FOOTBRIDGE),y)
44   ZTEXTADDR   = 0x00008000
45   PARAMS_PHYS     = 0x00000100
```

```
46  INITRD_PHYS     = 0x00800000
47  INITRD_VIRT     = 0xc0800000
48  endif
49
50  ifeq ($(CONFIG_ARCH_INTEGRATOR),y)
51  ZTEXTADDR   = 0x00008000
52  PARAMS_PHYS     = 0x00000100
53  INITRD_PHYS     = 0x00800000
54  INITRD_VIRT     = 0xc0800000
55  endif
56
57  ifeq ($(CONFIG_ARCH_NEXUSPCI),y)
58  ZTEXTADDR   = 0x40008000
59  endif
60
61  ifeq ($(CONFIG_ARCH_L7200),y)
62  # RAM based kernel
63  #ZTEXTADDR  = 0xf0400000
64  #ZRELADDR   = 0xf0008000
65
66  # FLASH based kernel
67  ZTEXTADDR   = 0x00010000
68  ZRELADDR    = 0xf0008000
69  ZBSSADDR    = 0xf03e0000
70  endif
71
72  ifeq ($(CONFIG_ARCH_P720T),y)
73  ZTEXTADDR   = 0xc0018000
74  PARAMS_PHYS     = 0xc0000100
75  INITRD_PHYS     = 0xc0400000
76  INITRD_VIRT     = 0xc0400000
77  endif
78
79  ifeq ($(CONFIG_ARCH_SA1100),y)
80  ZTEXTADDR   = 0xc0008000
81  ZRELADDR    = 0xc0008000
82  ifeq ($(CONFIG_SA1100_VICTOR),y)
83    ZTEXTADDR     = 0x00002000
84    ZBSSADDR  = 0xc0200000
85  endif
86  ifeq ($(CONFIG_SA1100_SHERMAN),y)
87    ZTEXTADDR      = 0x00050000
88    ZBSSADDR       = 0xc0200000
89  endif
90  ifeq ($(CONFIG_SA1100_GRAPHICSCLIENT),y)
91    ZTEXTADDR     = 0xC0200000
92  endif
```

```
93   ifeq ($(CONFIG_SA1100_YOPY),y)
94     ZTEXTADDR      = 0x00080000
95     ZBSSADDR       = 0xc0200000
96   endif
97   ifeq ($(CONFIG_SA1111),y)
98     ZRELADDR  = 0xc0208000
99   endif
100   endif
101
102   ifeq ($(CONFIG_ARCH_ANAKIN),y)
103   ZTEXTADDR  = 0x20008000
104   endif
105
106   ifeq ($(CONFIG_ARCH_ATMEL),y)
107   ZTEXTADDR = 0x1200000
108   ZRELADDR  = 0x1000000
109   endif
110
111   ifeq ($(CONFIG_BOARD_SNDS100),y)
112   ZRELADDR   = 0x00008000
113   ZTEXTADDR  = 0x00000000
114   endif
115
116   ifeq ($(CONFIG_ARCH_S3C44B0),y)
117   ZRELADDR   = 0x0C008000
118   ZTEXTADDR  = 0x00010000
119   ZBSSADDR   = 0x0c400000
120   endif
121
122   #
123   # If you don't define ZRELADDR above,
124   # then it defaults to ZTEXTADDR
125   #
126   ifeq ($(ZRELADDR),)
127   ZRELADDR  = $(ZTEXTADDR)
128   endif
129
130   export    SYSTEM   ZTEXTADDR   ZBSSADDR   ZRELADDR   INITRD_PHYS   INITRD_VIRT
PARAMS_PHYS
131
132   Image:    $(CONFIGURE) $(SYSTEM)
133     $(OBJCOPY) -O binary -R .note -R .comment -S $(SYSTEM) $@
134
135   bzImage: zImage
136
137   zImage:   $(CONFIGURE) compressed/$(LINUX)
138     $(OBJCOPY) -O binary -R .note -R .comment -S compressed/$(LINUX) $@
```

```
139
140   bootpImage: bootp/bootp
141     $(OBJCOPY) -O binary -R .note -R .comment -S bootp/bootp $@
142
143   compressed/$(LINUX): $(TOPDIR)/$(LINUX) dep
144     @$(MAKE) -C compressed $(LINUX)
145
146   bootp/bootp: zImage initrd
147     @$(MAKE) -C bootp bootp
148
149   initrd:
150    @test "$(INITRD_VIRT)" != "" || (echo This architecture does not support INITRD;
exit -1)
151     @test "$(INITRD)" != "" || (echo You must specify INITRD; exit -1)
152
153   install: $(CONFIGURE) Image
154     sh ./install.sh $(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION) Image
$(TOPDIR)/System.map "$(INSTALL_PATH)"
155
156   zinstall: $(CONFIGURE) zImage
157     sh ./install.sh $(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION) zImage
$(TOPDIR)/System.map "$(INSTALL_PATH)"
158
159   clean:
160     $(RM) Image zImage bootpImage
161     @$(MAKE) -C compressed clean
162     @$(MAKE) -C bootp clean
163
164   dep:
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/Makefile

```
1   #
2   # linux/arch/arm/boot/compressed/Makefile
3   #
4   # create a compressed vmlinuz image from the original vmlinux
5   #
6   # Note! SYSTEM, ZTEXTADDR, ZBSSADDR and ZRELADDR are now exported
7   # from arch/arm/boot/Makefile
8   #
9
10   HEAD        = head.o
11   OBJS        = misc.o
12
13   CFLAGS      =  $(CPPFLAGS)  -O2  -DSTDC_HEADERS  $(CFLAGS_PROC)  -msoft-float
-D__linux__
14
15   FONTC       = $(TOPDIR)/drivers/video/font_acorn_8x8.c
16   ZLDFLAGS    = -p -X -T $(LINUX).lds
17
18   #
19   # Architecture dependencies
20   #
21   ifeq ($(CONFIG_ARCH_ACORN),y)
22   OBJS       += ll_char_wr.o font.o
23   ZLDFLAGS   += -defsym params=$(PARAMS_PHYS)
24   endif
25
26   ifeq ($(CONFIG_ARCH_NETWINDER),y)
27   OBJS       += head-netwinder.o
28   endif
29
30   ifeq ($(CONFIG_ARCH_SHARK),y)
31   OBJS       += head-shark.o ofw-shark.o
32   endif
33
34   ifeq ($(CONFIG_ARCH_INTEGRATOR),y)
35   OBJS       += head-integrator.o
36   endif
37
38   ifeq ($(CONFIG_ARCH_FTVPCI),y)
39   OBJS       += head-ftvpci.o
40   endif
41
42   ifeq ($(CONFIG_ARCH_L7200),y)
43   OBJS       += head-l7200.o
44   endif
```

```
45
46  ifeq ($(CONFIG_ARCH_CLPS7500),y)
47  HEAD       = head-clps7500.o
48  endif
49
50  ifeq ($(CONFIG_ARCH_P720T),y)
51  # Borrow this code from SA1100
52  OBJS       += head-sa1100.o
53  endif
54
55  ifeq ($(CONFIG_ARCH_SA1100),y)
56  OBJS       += head-sa1100.o setup-sa1100.o
57  ifeq ($(CONFIG_SA1100_NANOENGINE),y)
58    OBJS += hw-bse.o
59  endif
60  endif
61
62  ifeq ($(CONFIG_ARCH_S3C44B0),y)
63  HEAD       = head.o
64  endif
65  SEDFLAGS   = s/TEXT_START/$(ZTEXTADDR)/;s/LOAD_ADDR/$(ZRELADDR)/;
66
67  ifneq ($(ZBSSADDR),)
68  SEDFLAGS   += s/BSS_START/$(ZBSSADDR)/
69  else
70  SEDFLAGS   += s/BSS_START/ALIGN(4)/
71  endif
72
73  all:       $(LINUX)
74
75  $(LINUX):  $(HEAD) $(OBJS) piggy.o $(LINUX).lds
76         $(LD) $(ZLDFLAGS) $(HEAD) $(OBJS) piggy.o -o $(LINUX)
77
78  $(HEAD):   $(HEAD:.o=.S)
79         $(CC) $(AFLAGS) -traditional -c $(HEAD:.o=.S)
80
81  piggy.o:   $(SYSTEM)
82         $(OBJCOPY) -O binary -R .note -R .comment -S $(SYSTEM) piggy
83         gzip $(GZFLAGS) < piggy > piggy.gz
84         $(LD) -r -o $@ -b binary piggy.gz
85         rm -f piggy piggy.gz
86
87  font.o:        $(FONTC)
88         $(CC) $(CFLAGS) -Dstatic= -c -o $@ $(FONTC)
89
90  $(LINUX).lds:  $(LINUX).lds.in
91         @sed "$(SEDFLAGS)" < $(LINUX).lds.in > $@
```

```
92
93  clean:;         rm -f $(LINUX) core piggy* $(LINUX).lds
94
95  .PHONY:    $(LINUX).lds clean
96
97  misc.o: misc.c $(TOPDIR)/include/asm/arch/uncompress.h $(TOPDIR)/lib/inflate.c
98
99  %.o: %.S
100     $(CC) $(AFLAGS) $(EXTRA_AFLAGS) $(AFLAGS_$@) -c -o $@ $<
```

uClinux-dist/linux-2.4.x/arch/armnommu/boot/compressed/head.S

```
1   /*
2    *  linux/arch/arm/boot/compressed/head.S
3    *
4    *  Copyright (C) 1996-1999 Russell King
5    *
6    * This program is free software; you can redistribute it and/or modify
7    * it under the terms of the GNU General Public License version 2 as
8    * published by the Free Software Foundation.
9    */
10   #include <linux/config.h>
11   #include <linux/linkage.h>
12
13   #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
14   #include <asm/hardware.h>
15   #endif
16
17   /*
18    * Debugging stuff
19    *
20    * Note that these macros must not contain any code which is not
21    * 100% relocatable.  Any attempt to do so will result in a crash.
22    * Please select one of the following when turning on debugging.
23    */
24   #ifdef DEBUG
25   #if 0 /* DC21285-type */
26          .macro  loadsp, rb
27          mov \rb, #0x7c000000
28          .endm
29          .macro  writeb, rb
30          strb    \rb, [r3, #0x3f8]
31          .endm
32   #elif 0 /* RiscPC-type */
33          .macro  loadsp, rb
34          mov \rb, #0x03000000
35          orr \rb, \rb, #0x00010000
36          .endm
37          .macro  writeb, rb
38          strb    \rb, [r3, #0x3f8 << 2]
39          .endm
40   #elif 0 /* integrator-type */
41          .macro  loadsp, rb
42          mov \rb, #0x16000000
43          .endm
44          .macro  writeb, rb
45          strb    \rb, [r3, #0]
```

```
46          .endm
47      #else
48      #error no serial architecture defined
49      #endif
50      #endif
51
52          .macro  kputc,val
53          mov r0, \val
54          bl  putc
55          .endm
56
57          .macro  kphex,val,len
58          mov r0, \val
59          mov r1, #\len
60          bl  phex
61          .endm
62
63          .macro  debug_reloc_start
64      #ifdef DEBUG
65          kputc   #'\n'
66          kphex   r6, 8       /* processor id */
67          kputc   #':'
68          kphex   r7, 8       /* architecture id */
69          kputc   #':'
70          mrc p15, 0, r0, c1, c0
71          kphex   r0, 8       /* control reg
72          kputc   #'\n'
73          kphex   r5, 8       /* decompressed kernel start */
74          kputc   #'-'
75          kphex   r8, 8       /* decompressed kernel end  */
76          kputc   #'>'
77          kphex   r4, 8       /* kernel execution address */
78          kputc   #'\n'
79      #endif
80          .endm
81
82          .macro  debug_reloc_end
83      #ifdef DEBUG
84          kphex   r5, 8       /* end of kernel */
85          kputc   #'\n'
86          mov r0, r4
87          bl  memdump     /* dump 256 bytes at start of kernel */
88      #endif
89          .endm
90
91          .section ".start", #alloc, #execinstr
92      /*
```

```
93    * sort out different calling conventions
94    */
95          .align
96    start:
97          .type   start,#function
98          .rept   8
99          mov r0, r0
100         .endr
101
102         b   1f
103         .word   0x016f2818      @ Magic numbers to help the loader
104         .word   start           @ absolute load/run zImage address
105         .word   _edata          @ zImage end address
106   1:        mov r7, r1          @ save architecture ID
107         mov r8, #0          @ save r0
108
109   #ifndef __ARM_ARCH_2__
110         /*
111          * Booting from Angel - need to enter SVC mode and disable
112          * FIQs/IRQs (numeric definitions from angel arm.h source).
113          * We only do this if we were in user mode on entry.
114          */
115         mrs r0, cpsr        @ get current mode
116         tst r0, #3          @ not user?
117         bne not_angel
118         mov r0, #0x17       @ angel_SWIreason_EnterSVC
119         swi 0x123456        @ angel_SWI_ARM
120   not_angel:
121         mrs r0, cpsr        @ turn off interrupts to
122         orr r0, r0, #0xc0       @ prevent angel from running
123         msr cpsr_c, r0
124   #else
125         teqp    pc, #0x0c000003     @ turn off interrupts
126   #endif
127
128         /*
129          * Note that some cache flushing and other stuff may
130          * be needed here - is there an Angel SWI call for this?
131          */
132
133         /*
134          * some architecture specific code can be inserted
135          * by the linker here, but it should preserve r7 and r8.
136          */
137
138         .text
139   #ifdef CONFIG_BOARD_SNDS100
```

```
140         /* set system parameters */
141         ldr r0, =SYSCFG
142         ldr r1, =rSYSCFG
143         str r1, [r0]
144
145         /* set reset memory map */
146         adr r0, SDRAM_SYSINIT_RESET
147         ldmia   r0, {r1-r12}
148         ldr r0, =SYS_INIT_BASE
149         stmia   r0, {r1-r12}
150
151         /* light led */
152         ldr r0, =IOPMOD
153         ldr r1, =0xFF
154         str r1, [r0]
155
156         ldr r0, =IOPDATA
157         ldr r1, =0xEF
158         str r1, [r0]
159
160         /* copy image to ram */
161         ldr r0, =0x0
162         ldr r1, =0x200000       /* 2M */
163         ldr r2, =0x1000000
164
165   rom2ram_copy_loop:
166         ldr r3, [r0], #4
167         str r3, [r2], #4
168         subs    r1, r1, #4
169         bne rom2ram_copy_loop
170
171         /* set boot memroy map */
172         adr r0, SDRAM_SYSINIT_BOOT
173         ldmia   r0, {r1-r12}
174         ldr r0, =SYS_INIT_BASE
175         stmia   r0, {r1-r12}
176   #endif
177
178   1:        adr r2, LC0
179         ldmia   r2, {r2, r3, r4, r5, sp}
180
181         mov r0, #0
182   1:        str r0, [r2], #4        @ clear bss
183         str r0, [r2], #4
184         str r0, [r2], #4
185         str r0, [r2], #4
186         cmp r2, r3
```

```
187         blt 1b
188
189     #ifdef CONFIG_CPU_WITH_CACHE
190     #ifndef CONFIG_BOARD_SNDS100
191     #ifndef CONFIG_ARCH_S3C44B0
192         mrc p15, 0, r6, c0, c0  @ get processor ID
193         bl  cache_on
194     #endif
195     #endif
196
197     #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
198         /*  cache_on    */
199         ldr r0, =SYSCFG
200         ldr r2, [r0]
201         orr r2, r2, #6
202         str r2, [r0]
203     #endif
204     #endif
205
206         mov r1, sp          @ malloc space above stack
207         add r2, sp, #0x10000    @ 64k max
208
209     #ifndef CONFIG_BOARD_SNDS100
210     #ifndef CONFIG_ARCH_S3C44B0
211         teq r4, r5          @ will we overwrite ourselves?
212         moveq   r5, r2          @ decompress after image
213         movne   r5, r4          @ decompress to final location
214     #endif
215     #endif
216     #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
217         mov r5, r2          @ decompress after image
218     #endif
219         mov r0, r5
220         mov r3, r7          @ r7 is arch id
221         bl  SYMBOL_NAME(decompress_kernel)
222
223         teq r4, r5          @ do we need to relocate
224         beq call_kernel     @ the kernel?
225
226         add r0, r0, #127
227         bic r0, r0, #127        @ align the kernel length
228     /*
229      * r0     = decompressed kernel length
230      * r1-r3  = unused
231      * r4     = kernel execution address
232      * r5     = decompressed kernel start
233      * r6     = processor ID
```

```
234     * r7     = architecture ID
235     * r8-r14 = unused
236     */
237          add r1, r5, r0      @ end of decompressed kernel
238          adr r2, reloc_start
239          adr r3, reloc_end
240     1:       ldmia   r2!, {r8 - r13}     @ copy relocation code
241          stmia   r1!, {r8 - r13}
242          ldmia   r2!, {r8 - r13}
243          stmia   r1!, {r8 - r13}
244          cmp r2, r3
245          blt 1b
246
247     #ifdef CONFIG_CPU_WITH_CACHE
248          bl  cache_clean_flush
249     #endif
250          add pc, r5, r0      @ call relocation code
251
252          .type   LC0, #object
253     LC0:      .word   __bss_start
254          .word   _end
255          .word   _load_addr
256          .word   _start
257          .word   user_stack+4096
258          .size   LC0, . - LC0
259
260     #ifdef CONFIG_BOARD_SNDS100
261     SDRAM_SYSINIT_RESET:  .word   rEXTDBWTH
262               .word   rROMCON0_R
263               .word   rROMCON1
264               .word   rROMCON2
265               .word   rROMCON3
266               .word   rROMCON4
267               .word   rROMCON5
268               .word   rSDRAMCON0_R
269               .word   rSDRAMCON1
270               .word   rSDRAMCON2
271               .word   rSDRAMCON3
272               .word   rREFEXTCON
273
274     SDRAM_SYSINIT_BOOT:   .word   rEXTDBWTH
275               .word   rROMCON0_B
276               .word   rROMCON1
277               .word   rROMCON2
278               .word   rROMCON3
279               .word   rROMCON4
280               .word   rROMCON5
```

```
281             .word   rSDRAMCON0_B
282             .word   rSDRAMCON1
283             .word   rSDRAMCON2
284             .word   rSDRAMCON3
285             .word   rREFEXTCON
286     #endif
287
288     /*
289      * Turn on the cache.  We need to setup some page tables so that we
290      * can have both the I and D caches on.
291      *
292      * We place the page tables 16k down from the kernel execution address,
293      * and we hope that nothing else is using it.  If we're using it, we
294      * will go pop!
295      *
296      * On entry,
297      *  r4 = kernel execution address
298      *  r6 = processor ID
299      *  r7 = architecture number
300      *  r8 = run-time address of "start"
301      * On exit,
302      *  r0, r1, r2, r3, r8, r9 corrupted
303      * This routine must preserve:
304      *  r4, r5, r6, r7
305      */
306          .align  5
307     cache_on: ldr r1, proc_sa110_type
308           eor r1, r1, r6
309           movs    r1, r1, lsr #5      @ catch SA110 and SA1100
310           beq 1f
311           ldr     r1, proc_sa1110_type
312           eor r1, r1, r6
313           movs    r1, r1, lsr #4
314           movne   pc, lr
315     1:
316           sub r3, r4, #16384      @ Page directory size
317           bic r3, r3, #0xff       @ Align the pointer
318           bic r3, r3, #0x3f00
319     /*
320      * Initialise the page tables, turning on the cacheable and bufferable
321      * bits for the RAM area only.
322      */
323           mov r0, r3
324           mov r8, r0, lsr #18
325           mov r8, r8, lsl #18     @ start of RAM
326           add r9, r8, #0x10000000 @ a reasonable RAM size
327           mov r1, #0x12
```

```
328         orr r1, r1, #3 << 10
329         add r2, r3, #16384
330   1:        cmp r1, r8          @ if virt > start of RAM
331         orrge   r1, r1, #0x0c       @ set cacheable, bufferable
332         cmp r1, r9          @ if virt > end of RAM
333         bicge   r1, r1, #0x0c       @ clear cacheable, bufferable
334         str r1, [r0], #4        @ 1:1 mapping
335         add r1, r1, #1048576
336         teq r0, r2
337         bne 1b
338   /*
339    * If ever we are running from Flash, then we surely want the cache
340    * to be enabled also for our execution instance...  We map 2MB of it
341    * so there is no map overlap problem for up to 1 MB compressed kernel.
342    * If the execution is in RAM then we would only be duplicating the above.
343    */
344         mov r1, #0x1e
345         orr r1, r1, #3 << 10
346         mov r2, pc, lsr #20
347         orr r1, r1, r2, lsl #20
348         add r0, r3, r2, lsl #2
349         str r1, [r0], #4
350         add r1, r1, #1048576
351         str r1, [r0]
352
353         mov r0, #0
354         mcr p15, 0, r0, c7, c10, 4  @ drain write buffer
355         mcr p15, 0, r0, c8, c7  @ flush I,D TLBs
356         mcr p15, 0, r3, c2, c0  @ load page table pointer
357         mov r0, #-1
358         mcr p15, 0, r0, c3, c0  @ load domain access register
359         mrc p15, 0, r0, c1, c0
360         orr r0, r0, #0x1000     @ I-cache enable
361   #ifndef DEBUG
362         orr r0, r0, #0x003d     @ Write buffer, mmu
363   #endif
364         mcr p15, 0, r0, c1, c0
365         mov pc, lr
366
367   /*
368    * This code is relocatable.  It is relocated by the above code to the end
369    * of the kernel and executed there.  During this time, we have no stacks.
370    *
371    * r0     = decompressed kernel length
372    * r1-r3  = unused
373    * r4     = kernel execution address
374    * r5     = decompressed kernel start
```

```
375     * r6     = processor ID
376     * r7     = architecture ID
377     * r8-r14 = unused
378     */
379          .align  5
380    reloc_start:  add r8, r5, r0
381          debug_reloc_start
382          mov r1, r4
383    1:
384          .rept   4
385          ldmia   r5!, {r0, r2, r3, r9 - r13} @ relocate kernel
386          stmia   r1!, {r0, r2, r3, r9 - r13}
387          .endr
388
389          cmp r5, r8
390          blt 1b
391          debug_reloc_end
392
393    call_kernel:
394    #ifdef CONFIG_CPU_WITH_CACHE
395          bl  cache_clean_flush
396          bl  cache_off
397    #endif
398          mov r0, #0
399          mov r1, r7          @ restore architecture number
400          mov pc, r4          @ call kernel
401
402    /*
403     * Here follow the relocatable cache support functions for
404     * the various processors.
405     */
406
407          .type   proc_sa110_type,#object
408    proc_sa110_type:
409          .word   0x4401a100
410          .size   proc_sa110_type, . - proc_sa110_type
411
412          .type   proc_sa1110_type,#object
413    proc_sa1110_type:
414          .word   0x6901b110
415          .size   proc_sa1110_type, . - proc_sa1110_type
416
417
418    #ifdef CONFIG_CPU_WITH_CACHE
419    /*
420     * Turn off the Cache and MMU.  ARMv3 does not support
421     * reading the control register, but ARMv4 does.
```

```
422     *
423     * On entry,  r6 = processor ID
424     * On exit,   r0, r1 corrupted
425     * This routine must preserve: r4, r6, r7
426     */
427          .align  5
428    cache_off:
429    #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
430          ldr r0, =SYSCFG
431          ldr r1, [r0]
432          bic r1, r1, #6
433          str r1, [r0]
434          mov pc, lr
435    #endif
436    #ifndef CONFIG_BOARD_SNDS100
437    #ifndef CONFIG_ARCH_S3C44B0
438    #ifdef CONFIG_CPU_ARM610
439          eor r1, r6, #0x41000000
440          eor r1, r1, #0x00560000
441          bic r1, r1, #0x0000001f
442          teq r1, #0x00000600
443          mov r0, #0x00000060     @ ARM6 control reg.
444          beq __armv3_cache_off
445    #endif
446    #ifdef CONFIG_CPU_ARM710
447          eor r1, r6, #0x41000000
448          bic r1, r1, #0x00070000
449          bic r1, r1, #0x000000ff
450          teq r1, #0x00007000     @ ARM7
451          teqne   r1, #0x00007100     @ ARM710
452          mov r0, #0x00000070     @ ARM7 control reg.
453          beq __armv3_cache_off
454    #endif
455          mrc p15, 0, r0, c1, c0
456          bic r0, r0, #0x000d
457          mcr p15, 0, r0, c1, c0  @ turn MMU and cache off
458          mov r0, #0
459          mcr p15, 0, r0, c7, c7  @ invalidate whole cache v4
460          mcr p15, 0, r0, c8, c7  @ invalidate whole TLB v4
461          mov pc, lr
462
463    __armv3_cache_off:
464          mcr p15, 0, r0, c1, c0  @ turn MMU and cache off
465          mov r0, #0
466          mcr p15, 0, r0, c7, c0  @ invalidate whole cache v3
467          mcr p15, 0, r0, c5, c0  @ invalidate whole TLB v3
468          mov pc, lr
```

```
469  #endif
470  #endif
471  /*
472   * Clean and flush the cache to maintain consistency.
473   *
474   * On entry,
475   *  r6 = processor ID
476   * On exit,
477   *  r1, r2, r12 corrupted
478   * This routine must preserve:
479   *  r4, r6, r7
480   */
481        .align  5
482  cache_clean_flush:
483  #if defined(CONFIG_BOARD_SNDS100) || defined(CONFIG_ARCH_S3C44B0)
484        @off_cache
485        ldr r2, =SYSCFG
486        ldr r1, =0x08
487        str r1, [r2]
488
489        mov r1, #0
490        ldr r2, =0x10002000
491        add r12, r2, #0x2800
492  1:
493        str r1, [r2], #0x10
494        teq r2, r12
495        bne 1b
496        mov pc, lr
497  #endif
498  #ifndef CONFIG_BOARD_SNDS100
499  #ifndef CONFIG_ARCH_S3C44B0
500        ldr r1, proc_sa110_type
501        eor r1, r1, r6
502        movs    r1, r1, lsr #5      @ catch SA110 and SA1100
503        beq 1f
504        ldr r1, proc_sa1110_type
505        eor r1, r1, r6
506        movs    r1, r1, lsr #4
507        movne   pc, lr
508  1:
509        bic r1, pc, #31
510        add r2, r1, #32768
511  1:        ldr r12, [r1], #32      @ s/w flush D cache
512        teq r1, r2
513        bne 1b
514
515        mcr p15, 0, r1, c7, c7, 0   @ flush I cache
```

```
516         mcr p15, 0, r1, c7, c10, 4  @ drain WB
517         mov pc, lr
518   #endif
519   #endif
520   /*
521    * Various debugging routines for printing hex characters and
522    * memory, which again must be relocatable.
523    */
524   #ifdef DEBUG
525         .type   phexbuf,#object
526   phexbuf:  .space  12
527         .size   phexbuf, . - phexbuf
528
529   phex:     adr r3, phexbuf
530         mov r2, #0
531         strb    r2, [r3, r1]
532   1:        subs    r1, r1, #1
533         movmi   r0, r3
534         bmi puts
535         and r2, r0, #15
536         mov r0, r0, lsr #4
537         cmp r2, #10
538         addge   r2, r2, #7
539         add r2, r2, #'0'
540         strb    r2, [r3, r1]
541         b   1b
542
543   puts:     loadsp  r3
544   1:        ldrb    r2, [r0], #1
545         teq r2, #0
546         moveq   pc, lr
547   2:        writeb  r2
548         mov r1, #0x00020000
549   3:        subs    r1, r1, #1
550         bne 3b
551         teq r2, #'\n'
552         moveq   r2, #'\r'
553         beq 2b
554         teq r0, #0
555         bne 1b
556         mov pc, lr
557   putc:
558         mov r2, r0
559         mov r0, #0
560         loadsp  r3
561         b   2b
562
```

```
563   memdump:  mov r12, r0
564         mov r10, lr
565         mov r11, #0
566   2:        mov r0, r11, lsl #2
567         add r0, r0, r12
568         mov r1, #8
569         bl  phex
570         mov r0, #':'
571         bl  putc
572   1:        mov r0, #' '
573         bl  putc
574         ldr r0, [r12, r11, lsl #2]
575         mov r1, #8
576         bl  phex
577         and r0, r11, #7
578         teq r0, #3
579         moveq   r0, #' '
580         bleq    putc
581         and r0, r11, #7
582         add r11, r11, #1
583         teq r0, #7
584         bne 1b
585         mov r0, #'\n'
586         bl  putc
587         cmp r11, #64
588         blt 2b
589         mov pc, r10
590   #endif
591   #endif
592
593   reloc_end:
594
595         .align
596         .section    ".stack"
597   user_stack:   .space  4096
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/head-armv.S

```
1   /*
2    *  linux/arch/arm/kernel/head-armv.S
3    *
4    *  Copyright (C) 1994-1999 Russell King
5    *
6    * This program is free software; you can redistribute it and/or modify
7    * it under the terms of the GNU General Public License version 2 as
8    * published by the Free Software Foundation.
9    *
10    *  32-bit kernel startup code for all architectures
11    *
12    * added S3C4510 code by Mac Wang
13    * added S3C4530 code by Arcturus Networks Inc.
14    *
15    */
16   #include <linux/config.h>
17   #include <linux/linkage.h>
18
19   #include <asm/assembler.h>
20   #include <asm/mach-types.h>
21   #include <asm/mach/arch.h>
22   #include <asm/hardware.h>
23
24   #define K(a,b,c)    ((a) << 24 | (b) << 12 | (c))
25
26   /*
27    * We place the page tables 16K below TEXTADDR.  Therefore, we must make sure
28    * that TEXTADDR is correctly set.  Currently, we expect the least significant
29    * "short" to be 0x8000, but we could probably relax this restriction to
30    * TEXTADDR > PAGE_OFFSET + 0x4000
31    *
32    * Note that swapper_pg_dir is the virtual address of the page tables, and
33    * pgtbl gives us a position-independent reference to these tables.  We can
34    * do this because stext == TEXTADDR
35    *
36    * swapper_pg_dir, pgtbl and krnladr are all closely related.
37    */
38   //#if ! defined(CONFIG_ARCH_P52)
39   //#if (TEXTADDR & 0xffff) != 0x8000
40   //#error TEXTADDR must start at 0xXXXX8000
41   //#endif
42   //#endif
43
44   #ifndef CONFIG_UCLINUX
45          .globl  SYMBOL_NAME(swapper_pg_dir)
```

```
46          .equ    SYMBOL_NAME(swapper_pg_dir), TEXTADDR - 0x4000
47          .macro  pgtbl, reg, rambase
48          adr \reg, stext
49          sub \reg, \reg, #0x4000
50          .endm
51
52  /*
53   * Since the page table is closely related to the kernel start address, we
54   * can convert the page table base address to the base address of the section
55   * containing both.
56   */
57          .macro  krnladr, rd, pgtable, rambase
58          bic \rd, \pgtable, #0x000ff000
59          .endm
60  #endif
61
62  /*
63   *  Kernel startup entry point.
64   *
65   * The rules are:
66   *  r0      - should be 0
67   *  r1      - unique architecture number
68   *  MMU     - off
69   *  I-cache - on or off
70   *  D-cache - off
71   *
72   * See linux/arch/arm/tools/mach-types for the complete list of numbers
73   * for r1.
74   */
75
76
77  .section ".text.init",#alloc,#execinstr
78          .type   stext, #function
79  ENTRY(stext)
80          mov r12, r0
81  /*
82   * NOTE!  Any code which is placed here should be done for one of
83   * the following reasons:
84   *
85   *  1. Compatability with old production boot firmware (ie, users
86   *     actually have and are booting the kernel with the old firmware)
87   *     and therefore will be eventually removed.
88   *  2. Cover the case when there is no boot firmware.  This is not
89   *     ideal, but in this case, it should ONLY set r0 and r1 to the
90   *     appropriate value.
91   */
92  #if defined(CONFIG_ARCH_NETWINDER)
```

```
93   /*
94    * Compatability cruft for old NetWinder NeTTroms.  This
95    * code is currently scheduled for destruction in 2.5.xx
96    */
97          .rept   8
98          mov r0, r0
99          .endr
100
101         adr r2, 1f
102         ldmdb   r2, {r7, r8}
103         and r3, r2, #0xc000
104         teq r3, #0x8000
105         beq __entry
106         bic r3, r2, #0xc000
107         orr r3, r3, #0x8000
108         mov r0, r3
109         mov r4, #64
110         sub r5, r8, r7
111         b   1f
112
113         .word   _stext
114         .word   __bss_start
115
116  1:
117         .rept   4
118         ldmia   r2!, {r6, r7, r8, r9}
119         stmia   r3!, {r6, r7, r8, r9}
120         .endr
121         subs    r4, r4, #64
122         bcs 1b
123         movs    r4, r5
124         mov r5, #0
125         movne   pc, r0
126
127         mov r1, #MACH_TYPE_NETWINDER    @ (will go in 2.5)
128         mov r12, #2 << 24           @ scheduled for removal in 2.5.xx
129         orr r12, r12, #5 << 12
130  __entry:
131  #endif
132  #if defined(CONFIG_ARCH_L7200)
133  /*
134   * FIXME - No bootloader, so manually set 'r1' with our architecture number.
135   */
136         mov r1, #MACH_TYPE_L7200
137  #elif defined(CONFIG_ARCH_INTEGRATOR)
138         mov r1, #MACH_TYPE_INTEGRATOR
139  #elif defined(CONFIG_ARCH_P52)
```

```
140         mov r1, #MACH_TYPE_P52
141     #elif defined(CONFIG_ARCH_SWARM)
142         mov r1, #MACH_TYPE_SWARM
143     #elif defined(CONFIG_BOARD_SNDS100)
144         mov r1, #MACH_TYPE_SNDS100
145
146     #elif defined (CONFIG_BOARD_SMDK40100)
147         mov r1, #MACH_TYPE_S3C3410
148     #elif defined(CONFIG_ARCH_S3C44B0)
149         mov r1, #MACH_TYPE_S3C44B0
150     #endif
151
152         mov r0, #F_BIT | I_BIT | MODE_SVC   @ make sure svc mode
153         msr cpsr_c, r0          @ and all irqs disabled
154
155     #if defined(CONFIG_ARCH_ATMEL)
156
157           adr   r5, LC0
158           ldmia r5, {r5, r6, r8, r9, sp}            @ Setup stack
159
160           /*  Copy data sections to their new home.  */
161
162
163           /*  Clear BSS */
164           mov   r4, #0
165     1:    cmp   r5, r8
166           strcc r4, [r5],#4
167           bcc   1b
168
169     /* FIXME */
170     #if 0
171           /*  Put initial values into stack.  This would normally be done
172           by sched_init() in kernel/sched.c, but that would overwrite the
173           stack we're already using.  That would be bad.
174           */
175           mov r5, sp
176           sub r5, r5, #0x2000
177           ldr r4, L_STACK_MAGIC
178           str r4, [r5], #4
179           ldr r4, L_STACK_UNTOUCHED_MAGIC
180     1:    cmp r5, sp
181           strcc r4, [r5], #4
182           bcc 1b
183     #endif
184           /*  Pretend we know what our processor code is (for arm_id)   */
185
186     @@@   ldr     r2, =0x41000000
```

```
187   @@@   orr r2, r2, #0x7000     @ FIXME --> 0x41007000
188
189     ldr r2, L_AT91_SF_CIDR
190     ldr r2, [r2]        @ read processor id
191
192     str     r2, [r6]
193     mov     r2, #MACH_TYPE_ATMEL
194     str     r2, [r9]
195
196           mov fp, #0
197           b start_kernel
198
199   LC0:  .long   __bss_start
200           .long processor_id
201           .long _end
202     .long   __machine_arch_type
203           .long init_task_union+8192
204   #endif
205
206   #if defined(CONFIG_ARCH_S3C44B0)
207           adr   r5, LC0
208           ldmia r5, {r5, r6, r8, r9, sp}            @ Setup stack
209
210           /*  Clear BSS */
211           mov   r4, #0
212   1:      cmp   r5, r8
213           strcc r4, [r5],#4
214           bcc   1b
215
216     /*  Pretend we know what our processor code is (for arm_id)   */
217
218     ldr r2, S3C44B0_PROCESSOR_TYPE
219
220     str     r2, [r6]
221     mov     r2, #MACH_TYPE_S3C44B0
222     str     r2, [r9]
223
224     /*  Enable Cache 8K */
225     ldr r2, =SYSCFG
226     ldr r4, =0x0e
227     str r4, [r2]
228
229           mov   fp, #0
230           b start_kernel
231
232   LC0:  .long   __bss_start
233           .long processor_id
```

```
234          .long _end
235    .long   __machine_arch_type
236          .long init_task_union+8192
237
238   S3C44B0_PROCESSOR_TYPE:
239    .long   0x36366036
240   #endif
241   #if defined(CONFIG_BOARD_SNDS100)
242
243          adr   r5, LC0
244          ldmia r5, {r5, r6, r8, r9, sp}            @ Setup stack
245
246          /*  Copy data sections to their new home.  */
247
248
249          /*  Clear BSS */
250          mov   r4, #0
251   1:     cmp   r5, r8
252          strcc r4, [r5],#4
253          bcc   1b
254
255          /*  Pretend we know what our processor code is (for arm_id)   */
256
257    ldr r2, S3C4510B_PROCESSOR_TYPE
258
259    str     r2, [r6]
260    mov     r2, #MACH_TYPE_SNDS100
261    str     r2, [r9]
262
263          mov fp, #0
264          b start_kernel
265
266   LC0:  .long   __bss_start
267          .long processor_id
268          .long _end
269    .long   __machine_arch_type
270          .long init_task_union+8192
271
272   S3C4510B_PROCESSOR_TYPE:
273    .long   0x36365036
274   #endif
275
276         #if   defined(CONFIG_ARCH_SAMSUNG)   &&   defined(CONFIG_CPU_ARM710)
&& !(CONFIG_ARCH_S3C44B0)
277
278                  adr     r2, LC0
279
```

```
280   #if     defined(CONFIG_ROMKERNEL)
281                   @  Copy data sections to their new home
282                   ldmia   r2, {r4-r11,sp}
283                   cmp     r9, r11
284                   beq     2f
285   1:              cmp     r11, r10
286                   ldrcc   r3, [r9], #4
287                   strcc   r3, [r11], #4
288                   bcc     1b
289   #else
290                   ldmia   r2, {r4-r8,sp}
291   #endif
292
293   2:              mov     r3, #0
294   3:              cmp     r7, r8
295                   strcc   r3, [r7], #4
296                   bcc     3b
297
298   #if     defined(CONFIG_CPU_S3C3410)
299         ldr r3, S3C3410_PROCESSOR_TYPE
300         str r3, [r5]
301         mov r3, #MACH_TYPE_S3C3410
302         str r3, [r6]
303   #endif
304   #if     defined(CONFIG_CPU_S3C4530)
305                    ldr     r3, [r4]                @ read system configuration
register
306                   bic     r3, r3, #0x83FFFFFF     @ get product identifier
307                   str     r3, [r5]                @ store it
308   #endif
309   #if     defined(CONFIG_BOARD_EVS3C4530HEI)
310                   mov     r3, #MACH_TYPE_EVS3C4530HEI
311                   str     r3, [r6]
312   #endif
313
314                   mov     fp, #0
315                   b       start_kernel
316
317   #if defined(CONFIG_CPU_S3C3410)
318   S3C3410_PROCESSOR_TYPE:
319         .long   0x34103410
320   #endif
321
322   LC0:
323   #if             defined(CONFIG_CPU_S3C4530)
324                   .long   0x3ff0000               @ r4
325   #endif
```

```
326  #if             defined(CONFIG_CPU_S3C3410)
327                  .long   0x07ff0000              @ r4
328  #endif
329                  .long   processor_id            @ r5
330                  .long   __machine_arch_type     @ r6
331                  .long   __bss_start             @ r7
332                  .long   _end                    @ r8
333
334  #if     defined (CONFIG_ROMKERNEL)
335  #if     defined (CONFIG_BOARD_SMDK40100)
336                  .long   __init_begin            @ r9    (?) ### @todo
337  #else
338                  .long   _end_romfs              @ r9    (src)
339  #endif
340                  .long   _edata                  @ r10
341                  .long   __init_begin            @ r11  RAM start address (dst)
342  #endif  /* CONFIG_ROMKERNEL */
343                  .long   init_task_union+8192    @ sp
344
345  #endif  /* CONFIG_ARCH_SAMSUNG */
346
347
348        bl  __lookup_processor_type
349        teq r10, #0             @ invalid processor?
350        moveq   r0, #'p'            @ yes, error 'p'
351        beq __error
352        bl  __lookup_architecture_type
353        teq r7, #0              @ invalid architecture?
354        moveq   r0, #'a'            @ yes, error 'a'
355        beq __error
356  #ifndef CONFIG_UCLINUX
357        bl  __create_page_tables
358  #endif
359        adr lr, __ret           @ return address
360        add pc, r10, #12            @ initialise processor
361                              @ (return control reg)
362
363  __switch_data:    .long   __mmap_switched
364        .long   SYMBOL_NAME(compat)
365        .long   SYMBOL_NAME(__bss_start)
366        .long   SYMBOL_NAME(_end)
367        .long   SYMBOL_NAME(processor_id)
368        .long   SYMBOL_NAME(__machine_arch_type)
369        .long   SYMBOL_NAME(cr_alignment)
370        .long   SYMBOL_NAME(init_task_union)+8192
371
372  __ret:        ldr lr, __switch_data
```

```
373   #ifdef CONFIG_CPU_WITH_CACHE
374   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
375         mcr p15, 0, r0, c1, c0
376         mov r0, r0
377         mov r0, r0
378         mov r0, r0
379   # else
380   #  warning    "FIXME: Enable Cache, Other settings without MCR Instruction"
381         @ Note r0 is initialized suitably to enable Cache
382         @ and additional settings if any in proc_armX.S file
383         @ A possible code here
384         @ldr    r2, CACHE_CONTROL_MEM_LOCATION
385         @str    r0, [r2]
386   # endif
387   #endif
388         mov pc, lr
389
390         /*
391          * This code follows on after the page
392          * table switch and jump above.
393          *
394          * r0  = processor control register
395          * r1  = machine ID
396          * r9  = processor ID
397          */
398         .align  5
399   __mmap_switched:
400         adr r3, __switch_data + 4
401         ldmia   r3, {r2, r4, r5, r6, r7, r8, sp}@ r2 = compat
402                               @ sp = stack pointer
403         str r12, [r2]
404
405         mov fp, #0              @ Clear BSS (and zero fp)
406   1:        cmp r4, r5
407         strcc   fp, [r4],#4
408         bcc 1b
409
410         str r9, [r6]            @ Save processor ID
411         str r1, [r7]            @ Save machine type
412   #ifdef CONFIG_ALIGNMENT_TRAP
413         orr r0, r0, #2          @ ...........A.
414   #endif
415         bic r2, r0, #2          @ Clear 'A' bit
416         stmia   r8, {r0, r2}            @ Save control register values
417         b   SYMBOL_NAME(start_kernel)
418
419
```

```
420  #ifndef CONFIG_UCLINUX
421  /*
422   * Setup the initial page tables.  We only setup the barest
423   * amount which are required to get the kernel running, which
424   * generally means mapping in the kernel code.
425   *
426   * We only map in 4MB of RAM, which should be sufficient in
427   * all cases.
428   *
429   * r5 = physical address of start of RAM
430   * r6 = physical IO address
431   * r7 = byte offset into page tables for IO
432   * r8 = page table flags
433   */
434  __create_page_tables:
435        pgtbl   r4, r5              @ page table address
436
437        /*
438         * Clear the 16K level 1 swapper page table
439         */
440        mov r0, r4
441        mov r3, #0
442        add r2, r0, #0x4000
443  1:        str r3, [r0], #4
444        str r3, [r0], #4
445        str r3, [r0], #4
446        str r3, [r0], #4
447        teq r0, r2
448        bne 1b
449
450        /*
451         * Create identity mapping for first MB of kernel to
452         * cater for the MMU enable.  This identity mapping
453         * will be removed by paging_init()
454         */
455        krnladr r2, r4, r5          @ start of kernel
456        add r3, r8, r2          @ flags + kernel base
457        str r3, [r4, r2, lsr #18]       @ identity mapping
458
459        /*
460         * Now setup the pagetables for our kernel direct
461         * mapped region.  We round TEXTADDR down to the
462         * nearest megabyte boundary.
463         */
464        add r0, r4, #(TEXTADDR & 0xff000000) >> 18 @ start of kernel
465        add r0, r0, #(TEXTADDR & 0x00f00000) >> 18
466        str r3, [r0], #4            @ PAGE_OFFSET + 0MB
```

```
467         add r3, r3, #1 << 20
468         str r3, [r0], #4            @ PAGE_OFFSET + 1MB
469         add r3, r3, #1 << 20
470         str r3, [r0], #4            @ PAGE_OFFSET + 2MB
471         add r3, r3, #1 << 20
472         str r3, [r0], #4            @ PAGE_OFFSET + 3MB
473
474         /*
475          * Ensure that the first section of RAM is present.
476          * we assume that:
477          *  1. the RAM is aligned to a 256MB boundary
478          *  2. the kernel is executing in the same 256MB chunk
479          *     as the start of RAM.
480          */
481         bic r0, r0, #0x0ff00000 >> 18   @ round down
482         and r2, r5, #0xf0000000     @ round down
483         add r3, r8, r2          @ flags + rambase
484         str r3, [r0]
485
486         bic r8, r8, #0x0c           @ turn off cacheable
487                                 @ and bufferable bits
488   #ifdef CONFIG_DEBUG_LL
489         /*
490          * Map in IO space for serial debugging.
491          * This allows debug messages to be output
492          * via a serial console before paging_init.
493          */
494         add r0, r4, r7
495         rsb r3, r7, #0x4000 @ PTRS_PER_PGD*sizeof(long)
496         cmp r3, #0x0800
497         addge   r2, r0, #0x0800
498         addlt   r2, r0, r3
499         orr r3, r6, r8
500   1:        str r3, [r0], #4
501         add r3, r3, #1 << 20
502         teq r0, r2
503         bne 1b
504   #ifdef CONFIG_ARCH_NETWINDER
505         /*
506          * If we're using the NetWinder, we need to map in
507          * the 16550-type serial port for the debug messages
508          */
509         teq r1, #MACH_TYPE_NETWINDER
510         bne 1f
511         add r0, r4, #0x3fc0
512         mov r3, #0x7c000000
513         orr r3, r3, r8
```

```
514         str r3, [r0], #4
515         add r3, r3, #1 << 20
516         str r3, [r0], #4
517   1:
518   #endif
519   #endif
520   #ifdef CONFIG_ARCH_RPC
521         /*
522          * Map in screen at 0x02000000 & SCREEN2_BASE
523          * Similar reasons here - for debug.  This is
524          * only for Acorn RiscPC architectures.
525          */
526         add r0, r4, #0x80           @ 02000000
527         mov r3, #0x02000000
528         orr r3, r3, r8
529         str r3, [r0]
530         add r0, r4, #0x3600         @ d8000000
531         str r3, [r0]
532   #endif
533         mov pc, lr
534   #endif
535
536
537   /*
538    * Exception handling.  Something went wrong and we can't
539    * proceed.  We ought to tell the user, but since we
540    * don't have any guarantee that we're even running on
541    * the right architecture, we do virtually nothing.
542    * r0 = ascii error character:
543    *    a = invalid architecture
544    *    p = invalid processor
545    *    i = invalid calling convention
546    *
547    * Generally, only serious errors cause this.
548    */
549   __error:
550   #ifdef CONFIG_DEBUG_LL
551         mov r8, r0              @ preserve r0
552         adr r0, err_str
553         bl  printascii
554         mov r0, r8
555         bl  printch
556   #endif
557   #ifdef CONFIG_ARCH_RPC
558   /*
559    * Turn the screen red on a error - RiscPC only.
560    */
```

```
561         mov r0, #0x02000000
562         mov r3, #0x11
563         orr r3, r3, r3, lsl #8
564         orr r3, r3, r3, lsl #16
565         str r3, [r0], #4
566         str r3, [r0], #4
567         str r3, [r0], #4
568         str r3, [r0], #4
569   #endif
570   1:        mov r0, r0
571         b   1b
572
573   #ifdef CONFIG_DEBUG_LL
574   err_str:  .asciz  "\nError: "
575         .align
576   #endif
577
578   /*
579    * Read processor ID register (CP#15, CR0), and look up in the linker-built
580    * supported processor list.  Note that we can't use the absolute addresses
581    * for the __proc_info lists since we aren't running with the MMU on
582    * (and therefore, we are not in the correct address space).  We have to
583    * calculate the offset.
584    *
585    * Returns:
586    *    r5, r6, r7 corrupted
587    *    r8  = page table flags
588    *    r9  = processor ID
589    *    r10 = pointer to processor structure
590    */
591   __lookup_processor_type:
592         adr r5, 2f
593         ldmia   r5, {r7, r9, r10}
594         sub r5, r5, r10         @ convert addresses
595         add r7, r7, r5          @ to our address space
596         add r10, r9, r5
597   #ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
598         mrc p15, 0, r9, c0, c0      @ get processor id
599   #else
600   # warning     "FIXME: Get Processor ID without MCR Instruction"
601         @ A possible code
602         @ldr    r9, PROCESSOR_ID_MEM_LOCATION
603         @ldr    r9, [r9]
604   #endif
605   1:        ldmia   r10, {r5, r6, r8}       @ value, mask, mmuflags
606         and r6, r6, r9          @ mask wanted bits
607         teq r5, r6
```

```
608         moveq   pc, lr
609         add r10, r10, #36           @ sizeof(proc_info_list)
610         cmp r10, r7
611         blt 1b
612         mov r10, #0             @ unknown processor
613         mov pc, lr
614
615   /*
616    * Look in include/asm-arm/procinfo.h and arch/arm/kernel/arch.[ch] for
617    * more information about the __proc_info and __arch_info structures.
618    */
619   2:        .long   __proc_info_end
620         .long   __proc_info_begin
621         .long   2b
622         .long   __arch_info_begin
623         .long   __arch_info_end
624
625   /*
626    * Lookup machine architecture in the linker-build list of architectures.
627    * Note that we can't use the absolute addresses for the __arch_info
628    * lists since we aren't running with the MMU on (and therefore, we are
629    * not in the correct address space).  We have to calculate the offset.
630    *
631    *  r1 = machine architecture number
632    * Returns:
633    *  r2, r3, r4 corrupted
634    *  r5 = physical start address of RAM
635    *  r6 = physical address of IO
636    *  r7 = byte offset into page tables for IO
637    */
638   __lookup_architecture_type:
639         adr r4, 2b
640         ldmia   r4, {r2, r3, r5, r6, r7}    @ throw away r2, r3
641         sub r5, r4, r5          @ convert addresses
642         add r4, r6, r5          @ to our address space
643         add r7, r7, r5
644   1:        ldr r5, [r4]            @ get machine type
645         teq r5, r1
646         beq 2f
647         add r4, r4, #SIZEOF_MACHINE_DESC
648         cmp r4, r7
649         blt 1b
650         mov r7, #0              @ unknown architecture
651         mov pc, lr
652   2:        ldmib   r4, {r5, r6, r7}        @ found, get results
653         mov r7, r7, lsr #18         @ pagetable byte offset
654         mov pc, lr
```

```
655
656   L_AT91_SF_CIDR: .long 0xfff00000
657
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/entry-armv.S

```
1   /*
2    *  linux/arch/armnommu/kernel/entry-armv.S [Kernel 2.4.x]
3    *
4    *  Copyright (C) 1996,1997,1998 Russell King.
5    *  ARM700 fix by Matthew Godbolt (linux-user@willothewisp.demon.co.uk)
6    *
7    * This program is free software; you can redistribute it and/or modify
8    * it under the terms of the GNU General Public License version 2 as
9    * published by the Free Software Foundation.
10    *
11    *  Low-level vector interface routines
12    *
13    *  Note:  there is a StrongARM bug in the STMIA rn, {regs}^ instruction that
causes
14    *  it to save wrong values...  Be aware!
15    *
16    * added some NET+ARM code by Joe deBlaquiere  --rp
17    * added S3C4510 code by Mac Wang
18    * added S3C4530 code by Arcturus Networks Inc.
19    *
20    */
21   #include <linux/config.h>
22   #include "entry-header.S"
23
24   #ifdef IOC_BASE
25   /* IOC / IOMD based hardware */
26   #include <asm/hardware/iomd.h>
27
28           .equ    ioc_base_high, IOC_BASE & 0xff000000
29           .equ    ioc_base_low, IOC_BASE & 0x00ff0000
30           .macro  disable_fiq
31           mov r12, #ioc_base_high
32           .if ioc_base_low
33           orr r12, r12, #ioc_base_low
34           .endif
35           strb    r12, [r12, #0x38]   @ Disable FIQ register
36           .endm
37
38           .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
39           mov r4, #ioc_base_high      @ point at IOC
40           .if ioc_base_low
41           orr r4, r4, #ioc_base_low
42           .endif
43           ldrb    \irqstat, [r4, #IOMD_IRQREQB]   @ get high priority first
44           ldr \base, =irq_prio_h
```

```
45          teq \irqstat, #0
46      #ifdef IOMD_BASE
47          ldreqb  \irqstat, [r4, #IOMD_DMAREQ]    @ get dma
48          addeq   \base, \base, #256      @ irq_prio_h table size
49          teqeq   \irqstat, #0
50          bne 2406f
51      #endif
52          ldreqb  \irqstat, [r4, #IOMD_IRQREQA]   @ get low priority
53          addeq   \base, \base, #256      @ irq_prio_d table size
54          teqeq   \irqstat, #0
55      #ifdef IOMD_IRQREQC
56          ldreqb  \irqstat, [r4, #IOMD_IRQREQC]
57          addeq   \base, \base, #256      @ irq_prio_l table size
58          teqeq   \irqstat, #0
59      #endif
60      #ifdef IOMD_IRQREQD
61          ldreqb  \irqstat, [r4, #IOMD_IRQREQD]
62          addeq   \base, \base, #256      @ irq_prio_lc table size
63          teqeq   \irqstat, #0
64      #endif
65      2406:       ldrneb  \irqnr, [\base, \irqstat]   @ get IRQ number
66          .endm
67
68      /*
69       * Interrupt table (incorporates priority).  Please note that we
70       * rely on the order of these tables (see above code).
71       */
72          .macro  irq_prio_table
73      irq_prio_h:     .byte    0,  8,  9,  8,10,10,10,10,11,11,11,11,10,10,10,10
74          .byte   12,  8,  9,  8,10,10,10,10,11,11,11,11,10,10,10,10
75          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
76          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
77          .byte   14,14,14,14,10,10,10,10,11,11,11,11,10,10,10,10
78          .byte   14,14,14,14,10,10,10,10,11,11,11,11,10,10,10,10
79          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
80          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
81          .byte   15,15,15,15,10,10,10,10,11,11,11,11,10,10,10,10
82          .byte   15,15,15,15,10,10,10,10,11,11,11,11,10,10,10,10
83          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
84          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
85          .byte   15,15,15,15,10,10,10,10,11,11,11,11,10,10,10,10
86          .byte   15,15,15,15,10,10,10,10,11,11,11,11,10,10,10,10
87          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
88          .byte   13,13,13,13,10,10,10,10,11,11,11,11,10,10,10,10
89      #ifdef IOMD_BASE
90      irq_prio_d:     .byte    0,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
91          .byte   20,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
```

```
92          .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
93          .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
94          .byte   22,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
95          .byte   22,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
96          .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
97          .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
98          .byte   23,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
99          .byte   23,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
100         .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
101         .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
102         .byte   22,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
103         .byte   22,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
104         .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
105         .byte   21,16,17,16,18,16,17,16,19,16,17,16,18,16,17,16
106     #endif
107     irq_prio_l:   .byte    0,  0,  1,  0,  2,  2,  2,  2,  3,  3,  3,  3,  3,  3,  3,  3
108         .byte    4,  0,  1,  0,  2,  2,  2,  2,  3,  3,  3,  3,  3,  3,  3,  3
109         .byte    5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5
110         .byte    5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5
111         .byte    6,  6,  6,  6,  6,  6,  6,  6,  3,  3,  3,  3,  3,  3,  3,  3
112         .byte    6,  6,  6,  6,  6,  6,  6,  6,  3,  3,  3,  3,  3,  3,  3,  3
113         .byte    5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5
114         .byte    5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5
115         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
116         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
117         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
118         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
119         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
120         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
121         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
122         .byte    7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7
123     #ifdef IOMD_IRQREQC
124     irq_prio_lc:  .byte   24,24,25,24,26,26,26,26,27,27,27,27,27,27,27,27
125         .byte   28,24,25,24,26,26,26,26,27,27,27,27,27,27,27,27
126         .byte   29,29,29,29,29,29,29,29,29,29,29,29,29,29,29,29
127         .byte   29,29,29,29,29,29,29,29,29,29,29,29,29,29,29,29
128         .byte   30,30,30,30,30,30,30,30,27,27,27,27,27,27,27,27
129         .byte   30,30,30,30,30,30,30,30,27,27,27,27,27,27,27,27
130         .byte   29,29,29,29,29,29,29,29,29,29,29,29,29,29,29,29
131         .byte   29,29,29,29,29,29,29,29,29,29,29,29,29,29,29,29
132         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
133         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
134         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
135         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
136         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
137         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
138         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
```

```
139         .byte   31,31,31,31,31,31,31,31,31,31,31,31,31,31,31,31
140   #endif
141   #ifdef IOMD_IRQREQD
142   irq_prio_ld:  .byte   40,40,41,40,42,42,42,42,43,43,43,43,43,43,43,43
143         .byte   44,40,41,40,42,42,42,42,43,43,43,43,43,43,43,43
144         .byte   45,45,45,45,45,45,45,45,45,45,45,45,45,45,45,45
145         .byte   45,45,45,45,45,45,45,45,45,45,45,45,45,45,45,45
146         .byte   46,46,46,46,46,46,46,46,43,43,43,43,43,43,43,43
147         .byte   46,46,46,46,46,46,46,46,43,43,43,43,43,43,43,43
148         .byte   45,45,45,45,45,45,45,45,45,45,45,45,45,45,45,45
149         .byte   45,45,45,45,45,45,45,45,45,45,45,45,45,45,45,45
150         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
151         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
152         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
153         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
154         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
155         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
156         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
157         .byte   47,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47
158   #endif
159         .endm
160
161   #elif defined(CONFIG_ARCH_EBSA110)
162
163   #define IRQ_STAT      0xff000000  /* read */
164
165         .macro  disable_fiq
166         .endm
167
168         .macro  get_irqnr_and_base, irqnr, stat, base, tmp
169         mov \base, #IRQ_STAT
170         ldrb    \stat, [\base]          @ get interrupts
171         mov \irqnr, #0
172         tst \stat, #15
173         addeq   \irqnr, \irqnr, #4
174         moveq   \stat, \stat, lsr #4
175         tst \stat, #3
176         addeq   \irqnr, \irqnr, #2
177         moveq   \stat, \stat, lsr #2
178         tst \stat, #1
179         addeq   \irqnr, \irqnr, #1
180         moveq   \stat, \stat, lsr #1
181         tst \stat, #1           @ bit 0 should be set
182         .endm
183
184         .macro  irq_prio_table
185         .endm
```

```
186
187   #elif defined(CONFIG_ARCH_SHARK)
188
189         .macro  disable_fiq
190         .endm
191
192         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
193         mov r4, #0xe0000000
194         orr r4, r4, #0x20
195
196         mov \irqstat, #0x0C
197         strb    \irqstat, [r4]          @outb(0x0C, 0x20) /* Poll command */
198         ldrb    \irqnr, [r4]            @irq = inb(0x20) & 7
199         and \irqstat, \irqnr, #0x80
200         teq \irqstat, #0
201         beq 43f
202         and \irqnr, \irqnr, #7
203         teq \irqnr, #2
204         bne 44f
205   43:       mov \irqstat, #0x0C
206         strb    \irqstat, [r4, #0x80]       @outb(0x0C, 0xA0) /* Poll command */
207         ldrb    \irqnr, [r4, #0x80]     @irq = (inb(0xA0) & 7) + 8
208         and \irqstat, \irqnr, #0x80
209         teq \irqstat, #0
210         beq 44f
211         and \irqnr, \irqnr, #7
212         add \irqnr, \irqnr, #8
213   44:       teq \irqstat, #0
214         .endm
215
216         .macro  irq_prio_table
217         .endm
218
219   #elif defined(CONFIG_FOOTBRIDGE)
220   #include <asm/hardware/dec21285.h>
221
222         .macro  disable_fiq
223         .endm
224
225         .equ    dc21285_high, ARMCSR_BASE & 0xff000000
226         .equ    dc21285_low, ARMCSR_BASE & 0x00ffffff
227
228         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
229         mov r4, #dc21285_high
230         .if dc21285_low
231         orr r4, r4, #dc21285_low
232         .endif
```

```
233         ldr \irqstat, [r4, #0x180]      @ get interrupts
234
235         mov \irqnr, #IRQ_SDRAMPARITY
236         tst \irqstat, #IRQ_MASK_SDRAMPARITY
237         bne 1001f
238
239         tst \irqstat, #IRQ_MASK_UART_RX
240         movne   \irqnr, #IRQ_CONRX
241         bne 1001f
242
243         tst \irqstat, #IRQ_MASK_DMA1
244         movne   \irqnr, #IRQ_DMA1
245         bne 1001f
246
247         tst \irqstat, #IRQ_MASK_DMA2
248         movne   \irqnr, #IRQ_DMA2
249         bne 1001f
250
251         tst \irqstat, #IRQ_MASK_IN0
252         movne   \irqnr, #IRQ_IN0
253         bne 1001f
254
255         tst \irqstat, #IRQ_MASK_IN1
256         movne   \irqnr, #IRQ_IN1
257         bne 1001f
258
259         tst \irqstat, #IRQ_MASK_IN2
260         movne   \irqnr, #IRQ_IN2
261         bne 1001f
262
263         tst \irqstat, #IRQ_MASK_IN3
264         movne   \irqnr, #IRQ_IN3
265         bne 1001f
266
267         tst \irqstat, #IRQ_MASK_PCI
268         movne   \irqnr, #IRQ_PCI
269         bne 1001f
270
271         tst \irqstat, #IRQ_MASK_DOORBELLHOST
272         movne   \irqnr, #IRQ_DOORBELLHOST
273         bne     1001f
274
275         tst \irqstat, #IRQ_MASK_I2OINPOST
276         movne   \irqnr, #IRQ_I2OINPOST
277         bne 1001f
278
279         tst \irqstat, #IRQ_MASK_TIMER1
```

```
280         movne   \irqnr, #IRQ_TIMER1
281         bne 1001f
282
283         tst \irqstat, #IRQ_MASK_TIMER2
284         movne   \irqnr, #IRQ_TIMER2
285         bne 1001f
286
287         tst \irqstat, #IRQ_MASK_TIMER3
288         movne   \irqnr, #IRQ_TIMER3
289         bne 1001f
290
291         tst \irqstat, #IRQ_MASK_UART_TX
292         movne   \irqnr, #IRQ_CONTX
293         bne 1001f
294
295         tst \irqstat, #IRQ_MASK_PCI_ABORT
296         movne   \irqnr, #IRQ_PCI_ABORT
297         bne 1001f
298
299         tst \irqstat, #IRQ_MASK_PCI_SERR
300         movne   \irqnr, #IRQ_PCI_SERR
301         bne 1001f
302
303         tst \irqstat, #IRQ_MASK_DISCARD_TIMER
304         movne   \irqnr, #IRQ_DISCARD_TIMER
305         bne 1001f
306
307         tst \irqstat, #IRQ_MASK_PCI_DPERR
308         movne   \irqnr, #IRQ_PCI_DPERR
309         bne 1001f
310
311         tst \irqstat, #IRQ_MASK_PCI_PERR
312         movne   \irqnr, #IRQ_PCI_PERR
313   1001:
314         .endm
315
316         .macro  irq_prio_table
317         .endm
318
319   #elif defined(CONFIG_ARCH_NEXUSPCI)
320
321         .macro  disable_fiq
322         .endm
323
324         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
325         ldr \irqstat, =INTCONT_BASE
326         ldr \base, =soft_irq_mask
```

```
327         ldr \irqstat, [\irqstat]        @ get interrupts
328         ldr \base, [\base]
329         mov \irqnr, #0
330         and \irqstat, \irqstat, \base   @ mask out disabled ones
331     1001:     tst \irqstat, #1
332         addeq   \irqnr, \irqnr, #1
333         moveq   \irqstat, \irqstat, lsr #1
334         tsteq   \irqnr, #32
335         beq 1001b
336         teq \irqnr, #32
337         .endm
338
339         .macro  irq_prio_table
340         .ltorg
341         .bss
342     ENTRY(soft_irq_mask)
343         .word   0
344         .text
345         .endm
346
347     #elif defined(CONFIG_ARCH_TBOX)
348
349         .macro  disable_fiq
350         .endm
351
352         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
353         ldr \irqstat, =0xffff7000
354         ldr \irqstat, [\irqstat]        @ get interrupts
355         ldr \base, =soft_irq_mask
356         ldr \base, [\base]
357         mov \irqnr, #0
358         and \irqstat, \irqstat, \base   @ mask out disabled ones
359     1001:     tst \irqstat, #1
360         addeq   \irqnr, \irqnr, #1
361         moveq   \irqstat, \irqstat, lsr #1
362         tsteq   \irqnr, #32
363         beq 1001b
364         teq \irqnr, #32
365         .endm
366
367         .macro  irq_prio_table
368         .ltorg
369         .bss
370     ENTRY(soft_irq_mask)
371         .word   0
372         .text
373         .endm
```

```
374
375   #elif defined(CONFIG_ARCH_SA1100)
376
377         .macro  disable_fiq
378         .endm
379
380         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
381         mov r4, #0xfa000000         @ ICIP = 0xfa050000
382         add r4, r4, #0x00050000
383         ldr \irqstat, [r4]          @ get irqs
384         ldr \irqnr, [r4, #4]        @ ICMR = 0xfa050004
385         ands    \irqstat, \irqstat, \irqnr
386         mov \irqnr, #0
387         beq 1001f
388         tst \irqstat, #0xff
389         moveq   \irqstat, \irqstat, lsr #8
390         addeq   \irqnr, \irqnr, #8
391         tsteq   \irqstat, #0xff
392         moveq   \irqstat, \irqstat, lsr #8
393         addeq   \irqnr, \irqnr, #8
394         tsteq   \irqstat, #0xff
395         moveq   \irqstat, \irqstat, lsr #8
396         addeq   \irqnr, \irqnr, #8
397         tst \irqstat, #0x0f
398         moveq   \irqstat, \irqstat, lsr #4
399         addeq   \irqnr, \irqnr, #4
400         tst \irqstat, #0x03
401         moveq   \irqstat, \irqstat, lsr #2
402         addeq   \irqnr, \irqnr, #2
403         tst \irqstat, #0x01
404         addeqs  \irqnr, \irqnr, #1
405   1001:
406         .endm
407
408         .macro  irq_prio_table
409         .endm
410
411   #elif defined(CONFIG_ARCH_L7200)
412
413         .equ    irq_base_addr,  IO_BASE_2
414
415         .macro  disable_fiq
416         .endm
417
418         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
419         mov     \irqstat, #irq_base_addr        @ Virt addr IRQ regs
420         add \irqstat, \irqstat, #0x00001000     @ Status reg
```

```
421         ldr     \irqstat, [\irqstat, #0]        @ get interrupts
422         mov     \irqnr, #0
423   1001:     tst     \irqstat, #1
424         addeq   \irqnr, \irqnr, #1
425         moveq   \irqstat, \irqstat, lsr #1
426         tsteq   \irqnr, #32
427         beq     1001b
428         teq     \irqnr, #32
429         .endm
430
431         .macro  irq_prio_table
432         .endm
433
434   #elif defined(CONFIG_ARCH_INTEGRATOR)
435
436         .macro  disable_fiq
437         .endm
438
439         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
440   /* FIXME: should not be using soo many LDRs here */
441         ldr \irqnr, =IO_ADDRESS(INTEGRATOR_IC_BASE)
442         ldr \irqstat, [\irqnr, #IRQ_STATUS]     @ get masked status
443         ldr \irqnr, =IO_ADDRESS(INTEGRATOR_HDR_BASE)
444         ldr \irqnr, [\irqnr, #(INTEGRATOR_HDR_IC_OFFSET+IRQ_STATUS)]
445         orr \irqstat, \irqstat, \irqnr, lsl #INTEGRATOR_CM_INT0
446
447         mov \irqnr, #0
448   1001:     tst \irqstat, #1
449         bne 1002f
450         add \irqnr, \irqnr, #1
451         mov \irqstat, \irqstat, lsr #1
452         cmp \irqnr, #22
453         bcc 1001b
454   1002:     /* EQ will be set if we reach 22 */
455         .endm
456
457         .macro  irq_prio_table
458         .endm
459
460   #elif defined(CONFIG_ARCH_CLPS711X)
461
462   #include <asm/hardware/clps7111.h>
463
464         .macro  disable_fiq
465         .endm
466
467   #if (INTSR2 - INTSR1) != (INTMR2 - INTMR1)
```

```
468  #error INTSR stride != INTMR stride
469  #endif
470
471         .macro  get_irqnr_and_base, irqnr, stat, base, mask
472         mov \base, #CLPS7111_BASE
473         ldr \stat, [\base, #INTSR1]
474         ldr \mask, [\base, #INTMR1]
475         mov \irqnr, #4
476         mov \mask, \mask, lsl #16
477         and \stat, \stat, \mask, lsr #16
478         movs    \stat, \stat, lsr #4
479         bne 1001f
480
481         add \base, \base, #INTSR2 - INTSR1
482         ldr \stat, [\base, #INTSR1]
483         ldr \mask, [\base, #INTMR1]
484         mov \irqnr, #16
485         mov \mask, \mask, lsl #16
486         and \stat, \stat, \mask, lsr #16
487
488  1001:     tst \stat, #255
489         addeq   \irqnr, \irqnr, #8
490         moveq   \stat, \stat, lsr #8
491         tst \stat, #15
492         addeq   \irqnr, \irqnr, #4
493         moveq   \stat, \stat, lsr #4
494         tst \stat, #3
495         addeq   \irqnr, \irqnr, #2
496         moveq   \stat, \stat, lsr #2
497         tst \stat, #1
498         addeq   \irqnr, \irqnr, #1
499         moveq   \stat, \stat, lsr #1
500         tst \stat, #1           @ bit 0 should be set
501         .endm
502
503         .macro  irq_prio_table
504         .endm
505
506  #elif defined(CONFIG_ARCH_ANAKIN)
507
508         .macro  disable_fiq
509         .endm
510
511         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
512         mov \base, #IO_BASE
513         mov \irqstat, #INTERRUPT_CONTROLLER
514         ldr \tmp, =anakin_irq_mask
```

```
515         ldr \irqstat, [\base, \irqstat]
516         ldr \tmp, [\tmp]
517         ands    \irqstat, \irqstat, \tmp
518         ldrne   \tmp, =anakin_active_irqs
519         strne   \irqstat, [\tmp]
520         movne   \irqnr, #IRQ_ANAKIN
521         .endm
522
523         .macro  irq_prio_table
524         .ltorg
525         .bss
526   ENTRY(anakin_irq_mask)
527         .word   0
528   ENTRY(anakin_active_irqs)
529         .space  4
530         .text
531         .endm
532
533   #elif defined(CONFIG_ARCH_CNXT)
534
535         .macro  disable_fiq
536         .endm
537
538
539         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
540
541   #ifdef CONFIG_ARCH_P52
542
543         ldr r4, =P52INT_STATUS_M
544         ldr \irqstat, [r4]
545
546
547         tst \irqstat, #P52INT_MASK_TIMER_1
548         movne   \irqnr, #P52INT_LVL_TIMER_1
549         bne 1001f
550
551         tst \irqstat, #P52INT_MASK_TIMER_2
552         movne   \irqnr, #P52INT_LVL_TIMER_2
553         bne 1001f
554
555         tst \irqstat, #P52INT_MASK_TIMER_3
556         movne   \irqnr, #P52INT_LVL_TIMER_3
557         bne 1001f
558
559         tst \irqstat, #P52INT_MASK_TIMER_4
560         movne   \irqnr, #P52INT_LVL_TIMER_4
561         bne 1001f
```

```
562
563         tst \irqstat, #P52INT_MASK_USB
564         movne   \irqnr, #P52INT_LVL_USB
565         bne 1001f
566
567         tst \irqstat, #P52INT_MASK_HOST
568         movne   \irqnr, #P52INT_LVL_HOST
569         bne 1001f
570
571         tst \irqstat, #P52INT_MASK_HOST_ERR
572         movne   \irqnr, #P52INT_LVL_HOST_ERR
573         bne 1001f
574
575         tst \irqstat, #P52INT_MASK_DMA8
576         movne   \irqnr, #P52INT_LVL_DMA8
577         bne 1001f
578
579         tst \irqstat, #P52INT_MASK_DMA6
580         movne   \irqnr, #P52INT_LVL_DMA6
581         bne     1001f
582
583         tst \irqstat, #P52INT_MASK_DMA5
584         movne   \irqnr, #P52INT_LVL_DMA5
585         bne 1001f
586
587         tst \irqstat, #P52INT_MASK_DMA4
588         movne   \irqnr, #P52INT_LVL_DMA4
589         bne 1001f
590
591         tst \irqstat, #P52INT_MASK_DMA3
592         movne   \irqnr, #P52INT_LVL_DMA3
593         bne 1001f
594
595         tst \irqstat, #P52INT_MASK_DMA2
596         movne   \irqnr, #P52INT_LVL_DMA2
597         bne 1001f
598
599         tst \irqstat, #P52INT_MASK_DMA1
600         movne   \irqnr, #P52INT_LVL_DMA1
601         bne 1001f
602
603         tst \irqstat, #P52INT_MASK_DMA_ERR
604         movne   \irqnr, #P52INT_LVL_DMA_ERR
605         bne 1001f
606
607         tst \irqstat, #P52INT_MASK_E2_ERR
608         movne   \irqnr, #P52INT_LVL_E2_ERR
```

```
609         bne 1001f
610
611         tst \irqstat, #P52INT_MASK_E1_ERR
612         movne   \irqnr, #P52INT_LVL_E1_ERR
613         bne 1001f
614
615         tst \irqstat, #P52INT_MASK_DSL
616         movne   \irqnr, #P52INT_LVL_DSL
617         bne 1001f
618
619         tst \irqstat, #P52INT_MASK_GPIO
620         movne   \irqnr, #P52INT_LVL_GPIO
621         bne 1001f
622
623         tst \irqstat, #P52INT_MASK_COMMTX
624         movne   \irqnr, #P52INT_LVL_COMMTX
625         bne 1001f
626
627         tst \irqstat, #P52INT_MASK_COMMRX
628         movne   \irqnr, #P52INT_LVL_COMMRX
629         bne 1001f
630
631         tst \irqstat, #P52INT_MASK_SW1
632         movne   \irqnr, #P52INT_LVL_SW1
633         bne 1001f
634
635         tst \irqstat, #P52INT_MASK_SW2
636         movne   \irqnr, #P52INT_LVL_SW2
637         bne 1001f
638
639         tst \irqstat, #P52INT_MASK_SW3
640         movne   \irqnr, #P52INT_LVL_SW3
641         bne 1001f
642
643         tst \irqstat, #P52INT_MASK_SW4
644         movne   \irqnr, #P52INT_LVL_SW4
645
646
647   1001:
648
649   #endif
650
651
652   #ifdef CONFIG_ARCH_SPIPE
653         @
654         @ not been tested
655         @
```

```
656         ldr r4, =0x350044
657         ldr \irqnr, [r4]
658
659         mov r5,#0
660     icloopusr:
661         cmp r5,#31
662         ble icshftusr
663         b icendusr
664     icshftusr:
665         mov r4,r6,lsr r5        @ shift untill we get one
666         and r0,r4,#1            @ int priority is in the status
667         cmp r0,#0
668         beq incicusr
669         b icendusr
670     incicusr:
671         add r5,r5,#1
672         b icloopusr
673
674     icendusr:
675         mov r0, r5
676     #endif
677
678         .endm
679
680         .macro  irq_prio_table
681         .endm
682
683     #elif defined(CONFIG_ARCH_ATMEL)
684
685         .macro  disable_fiq
686         .endm
687
688         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
689         ldr r4, =AIC_IVR
690         ldr \irqnr, [r4]        @ignore value
691         ldr r4, =AIC_ISR        @read interrupt nr.
692         ldr \irqnr, [r4]
693         teq \irqnr, #0
694
695         ldreq   r4, =AIC_EOICR      @ EOI
696         streq   r4, [r4]        @ value=dont care
697         .endm
698
699         .macro  irq_prio_table
700         .endm
701
702     #elif defined(CONFIG_ARCH_NETARM)
```

```
703  #include <asm/arch/netarm_gen_module.h>
704
705        .macro  disable_fiq
706        mrs r13, spsr
707        orr r13, r13, #F_BIT
708        msr spsr_c, r13
709        .endm
710
711        .macro  get_irqnr_and_base, irqnr, stat, base, temp
712        ldr     \irqnr, =(NETARM_GEN_MODULE_BASE+NETARM_GEN_INTR_STATUS_EN)
713
714        ldr \base, [\irqnr, #0] @ stash ISTATUS
715        mov \irqnr, #0
716  1001:
717        cmp \base, #0       @ no flags set?
718        beq 1002f           @ keep "eq" status when finishing!
719        tst \base, #1       @ lsb set?
720        addeq   \irqnr, \irqnr, #1  @ if not, incr IRQ#
721        moveq   \base, \base, LSR #1    @ r = r >> 1
722        beq 1001b
723
724        cmp \irqnr, #32     @ IRQ# too big?
725        blge    _netarm_led_FAIL2
726  1002:
727        adr \base, irq_prio_netarm
728        .endm
729
730        .macro  irq_prio_table
731  irq_prio_netarm:
732        .byte   0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
733        .byte   16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
734        .endm
735
736  #elif defined(CONFIG_CPU_S3C4510) || defined(CONFIG_CPU_S3C4530)
737
738        .macro disable_fiq
739        .endm
740
741        .macro get_irqnr_and_base, irqnr, irqstat, base, tmp
742        ldr \base, =INTOSET_IRQ
743        ldr \irqnr, [\base]
744        mov \irqnr, \irqnr, lsr #2
745        teq \irqnr, #NR_IRQS
746        .endm
747
748        .macro irq_prio_table
749        .endm
```

```
750
751   #elif defined(CONFIG_CPU_S3C3410)
752
753         .macro disable_fiq
754         .endm
755
756         .macro get_irqnr_and_base, irqnr, irqstat, base, tmp
757         ldr \irqstat, =.LCintpnd    @ INTPND
758         ldr \irqstat, [\irqstat]
759         ldr     \irqstat, [\irqstat]    @ interrupt pending register
760         mov     \irqnr, #0
761   1001:
762         tst     \irqstat, #1
763         bne     1002f
764         add     \irqnr, \irqnr, #1
765         mov     \irqstat, \irqstat, lsr #1
766         cmp     \irqnr, #32
767         bcc     1001b
768   1002:     /* EQ will be set if we reach 32 */
769         .endm
770
771         .macro irq_prio_table
772         .endm
773
774   #elif defined(CONFIG_ARCH_S3C44B0)
775         .macro disable_fiq
776         .endm
777         .macro get_irqnr_and_base, irqnr, irqstat, base, tmp
778         ldr \base, =INTPND
779         ldr \base, [\base]
780         mov \irqnr, #0
781   2222:
782         tst \base, #1
783         bne 1111f
784         add \irqnr, \irqnr, #1
785         mov \base, \base, lsr #1
786         cmp \irqnr, #26
787         bcc 2222b
788   1111:
789         .endm
790
791         .macro irq_prio_table
792         .endm
793
794   #elif defined(CONFIG_ARCH_SWARM)
795
796         .macro  disable_fiq
```

```
797         .endm
798
799         .macro  get_irqnr_and_base, irqnr, irqstat, base, tmp
800         ldr     \irqstat, =SWARM_INT_IRQ_STATUS
801         ldr     \irqstat, [\irqstat]            @ get interrupts
802
803         mov     \irqnr, #0
804   1001:
805         tst     \irqstat, #1
806         bne     1002f
807         add     \irqnr, \irqnr, #1
808         mov     \irqstat, \irqstat, lsr #1
809         cmp     \irqnr, #32
810         bcc     1001b
811   1002:     /* EQ will be set if we reach 32 */
812         .endm
813
814         .macro  irq_prio_table
815   irq_prio_swarm:
816         .byte
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
,31
817         .endm
818
819   #else
820   #error Unknown architecture
821   #endif
822
823   /*
824    * Invalid mode handlers
825    */
826   __pabt_invalid:   sub sp, sp, #S_FRAME_SIZE       @ Allocate frame size in one
go
827         stmia   sp, {r0 - lr}           @ Save XXX r0 - lr
828         ldr r4, .LCabt
829         mov r1, #BAD_PREFETCH
830         b   1f
831
832   __dabt_invalid:   sub sp, sp, #S_FRAME_SIZE
833         stmia   sp, {r0 - lr}           @ Save SVC r0 - lr [lr *should* be intact]
834         ldr r4, .LCabt
835         mov r1, #BAD_DATA
836         b   1f
837
838   __irq_invalid:    sub sp, sp, #S_FRAME_SIZE       @ Allocate space on stack
for frame
839         stmfd   sp, {r0 - lr}           @ Save r0 - lr
```

```
840         ldr r4, .LCirq
841         mov r1, #BAD_IRQ
842         b   1f
843
844   __und_invalid:    sub sp, sp, #S_FRAME_SIZE
845         stmia   sp, {r0 - lr}
846         ldr r4, .LCund
847         mov r1, #BAD_UNDEFINSTR     @ int reason
848
849   1:        zero_fp
850         ldmia   r4, {r5 - r7}           @ Get XXX pc, cpsr, old_r0
851         add r4, sp, #S_PC
852         stmia   r4, {r5 - r7}           @ Save XXX pc, cpsr, old_r0
853         mov r0, sp
854         and r2, r6, #31         @ int mode
855         b   SYMBOL_NAME(bad_mode)
856
857   #if defined CONFIG_FPE_NWFPE || defined CONFIG_FPE_FASTFPE
858         /* The FPE is always present */
859         .equ    fpe_not_present, 0
860   #else
861   wfs_mask_data:    .word   0x0e200110          @ WFS/RFS
862         .word   0x0fef0fff
863         .word   0x0d000100          @ LDF [sp]/STF [sp]
864         .word   0x0d000100          @ LDF [fp]/STF [fp]
865         .word   0x0f000f00
866
867   /* We get here if an undefined instruction happens and the floating
868    * point emulator is not present.  If the offending instruction was
869    * a WFS, we just perform a normal return as if we had emulated the
870    * operation.  This is a hack to allow some basic userland binaries
871    * to run so that the emulator module proper can be loaded. --philb
872    */
873   fpe_not_present:
874         adr r10, wfs_mask_data
875         ldmia   r10, {r4, r5, r6, r7, r8}
876         ldr r10, [sp, #S_PC]        @ Load PC
877         sub r10, r10, #4
878         mask_pc r10, r10
879         ldrt    r10, [r10]          @ get instruction
880         and r5, r10, r5
881         teq r5, r4              @ Is it WFS?
882         moveq   pc, r9
883         and r5, r10, r8
884         teq r5, r6              @ Is it LDF/STF on sp or fp?
885         teqne   r5, r7
886         movne   pc, lr
```

```
887         tst r10, #0x00200000        @ Does it have WB
888         moveq   pc, r9
889         and r4, r10, #255           @ get offset
890         and r6, r10, #0x000f0000
891         tst r10, #0x00800000        @ +/-
892         ldr r5, [sp, r6, lsr #14]       @ Load reg
893         rsbeq   r4, r4, #0
894         add r5, r5, r4, lsl #2
895         str r5, [sp, r6, lsr #14]       @ Save reg
896         mov pc, r9
897   #endif
898
899   /*
900    * SVC mode handlers
901    */
902         .align  5
903   __dabt_svc:   sub sp, sp, #S_FRAME_SIZE
904         stmia   sp, {r0 - r12}          @ save r0 - r12
905         ldr r2, .LCabt
906         add r0, sp, #S_FRAME_SIZE
907         ldmia   r2, {r2 - r4}           @ get pc, cpsr
908         add r5, sp, #S_SP
909         mov r1, lr
910         stmia   r5, {r0 - r4}           @ save sp_SVC, lr_SVC, pc, cpsr, old_ro
911         mrs r9, cpsr            @ Enable interrupts if they were
912         tst r3, #I_BIT
913         biceq   r9, r9, #I_BIT          @ previously
914         mov r0, r2
915   /*
916    * This routine must not corrupt r9
917    */
918   #ifdef MULTI_CPU
919         ldr r2, .LCprocfns
920         mov lr, pc
921         ldr pc, [r2]            @ call processor specific code
922   #else
923         bl  cpu_data_abort
924   #endif
925         msr cpsr_c, r9
926         mov r2, sp
927         bl  SYMBOL_NAME(do_DataAbort)
928         mrs r0, cpsr
929         bic r0, r0, #MODE_MASK
930         orr r0, r0, #I_BIT | MODE_SVC   @ preserve FIQ bit
931         msr cpsr_c, r0
932         ldr r0, [sp, #S_PSR]
933         msr spsr, r0
```

```
934         ldmia   sp, {r0 - pc}^          @ load r0 - pc, cpsr
935
936         .align  5
937   __irq_svc:    sub sp, sp, #S_FRAME_SIZE
938         stmia   sp, {r0 - r12}          @ save r0 - r1
939         ldr r7, .LCirq
940         add r5, sp, #S_FRAME_SIZE
941         ldmia   r7, {r7 - r9}
942         add r4, sp, #S_SP
943         mov r6, lr
944         stmia   r4, {r5, r6, r7, r8, r9}    @ save sp_SVC, lr_SVC, pc, cpsr,
old_ro
945   1:        get_irqnr_and_base r0, r6, r5, lr
946         movne   r1, sp
947         @
948         @ routine called with r0 = irq number, r1 = struct pt_regs *
949         @
950         adrsvc  ne, lr, 1b
951         bne do_IRQ
952         ldr r0, [sp, #S_PSR]        @ irqs are already disabled
953         msr spsr, r0
954         ldmia   sp, {r0 - pc}^          @ load r0 - pc, cpsr
955
956         .align  5
957   __und_svc:    sub sp, sp, #S_FRAME_SIZE
958         stmia   sp, {r0 - r12}          @ save r0 - r12
959         ldr r7, .LCund
960         mov r6, lr
961         ldmia   r7, {r7 - r9}
962         add r5, sp, #S_FRAME_SIZE
963         add r4, sp, #S_SP
964         stmia   r4, {r5 - r9}           @ save sp_SVC, lr_SVC, pc, cpsr, old_ro
965
966         adrsvc  al, r9, 1f          @ r9  = normal FP return
967         bl  call_fpe            @ lr  = undefined instr return
968
969         mov r0, r5              @ unsigned long pc
970         mov r1, sp              @ struct pt_regs *regs
971         bl  SYMBOL_NAME(do_undefinstr)
972
973   1:        mrs r0, cpsr
974         bic r0, r0, #MODE_MASK      @ preserve F-bit
975         orr r0, r0, #I_BIT | MODE_SVC
976         msr cpsr_c, r0
977         ldr lr, [sp, #S_PSR]        @ Get SVC cpsr
978         msr spsr, lr
979         ldmia   sp, {r0 - pc}^          @ Restore SVC registers
```

```
980
981         .align  5
982   __pabt_svc:   sub sp, sp, #S_FRAME_SIZE
983         stmia   sp, {r0 - r12}          @ save r0 - r12
984         ldr r2, .LCabt
985         add r0, sp, #S_FRAME_SIZE
986         ldmia   r2, {r2 - r4}           @ get pc, cpsr
987         add r5, sp, #S_SP
988         mov r1, lr
989         stmia   r5, {r0 - r4}           @ save sp_SVC, lr_SVC, pc, cpsr, old_ro
990         mrs r9, cpsr            @ Enable interrupts if they were
991         tst r3, #I_BIT
992         biceq   r9, r9, #I_BIT          @ previously
993         msr cpsr_c, r9
994         mov r0, r2              @ address (pc)
995         mov r1, sp              @ regs
996         bl  SYMBOL_NAME(do_PrefetchAbort)   @ call abort handler
997         mrs r0, cpsr
998         bic r0, r0, #MODE_MASK      @ preserve F-bit
999         orr r0, r0, #I_BIT | MODE_SVC
1000        msr cpsr_c, r0
1001        ldr r0, [sp, #S_PSR]
1002        msr spsr, r0
1003        ldmia   sp, {r0 - pc}^          @ load r0 - pc, cpsr
1004
1005        .align  5
1006  .LCirq:       .word   __temp_irq
1007  .LCund:       .word   __temp_und
1008  .LCabt:       .word   __temp_abt
1009  #ifdef MULTI_CPU
1010  .LCprocfns:   .word   SYMBOL_NAME(processor)
1011  #endif
1012  .LCfp:        .word   SYMBOL_NAME(fp_enter)
1013  #ifdef CONFIG_CPU_S3C3410
1014  .LCintpnd:    .word   S3C3410X_INTPND
1015  #endif
1016        irq_prio_table
1017
1018  /*
1019   * User mode handlers
1020   */
1021        .align  5
1022  __dabt_usr:   sub sp, sp, #S_FRAME_SIZE       @ Allocate frame size in one go
1023        stmia   sp, {r0 - r12}          @ save r0 - r12
1024        ldr r4, .LCabt
1025        add r3, sp, #S_PC
1026        ldmia   r4, {r0 - r2}           @ Get USR pc, cpsr
```

```
1027        stmia   r3, {r0 - r2}           @ Save USR pc, cpsr, old_r0
1028        stmdb   r3, {sp, lr}^
1029        alignment_trap r4, r7, __temp_abt
1030        zero_fp
1031    #ifdef MULTI_CPU
1032        ldr r2, .LCprocfns
1033        mov lr, pc
1034        ldr pc, [r2]            @ call processor specific code
1035    #else
1036        bl  cpu_data_abort
1037    #endif
1038        mrs r2, cpsr
1039        bic r2, r2, #I_BIT | MODE_MASK  @ preserve F-bit
1040        orr r2, r2, #MODE_SVC
1041        msr cpsr_c, r2          @ Enable interrupts
1042        mov r2, sp
1043        adrsvc  al, lr, ret_from_exception
1044        b   SYMBOL_NAME(do_DataAbort)
1045
1046        .align  5
1047    __irq_usr:   sub sp, sp, #S_FRAME_SIZE
1048        stmia   sp, {r0 - r12}          @ save r0 - r12
1049        ldr r4, .LCirq
1050        add r8, sp, #S_PC
1051        ldmia   r4, {r5 - r7}           @ get saved PC, SPSR
1052        stmia   r8, {r5 - r7}           @ save pc, psr, old_r0
1053        stmdb   r8, {sp, lr}^
1054        alignment_trap r4, r7, __temp_irq
1055        zero_fp
1056    1:      get_irqnr_and_base r0, r6, r5, lr
1057        movne   r1, sp
1058        adrsvc  ne, lr, 1b
1059        @
1060        @ routine called with r0 = irq number, r1 = struct pt_regs *
1061        @
1062        bne do_IRQ
1063        mov why, #0
1064        get_current_task tsk
1065        b   ret_to_user
1066
1067        .align  5
1068    __und_usr:   sub sp, sp, #S_FRAME_SIZE      @ Allocate frame size in one go
1069        stmia   sp, {r0 - r12}          @ Save r0 - r12
1070        ldr r4, .LCund
1071        add r8, sp, #S_PC
1072        ldmia   r4, {r5 - r7}
1073        stmia   r8, {r5 - r7}           @ Save USR pc, cpsr, old_r0
```

```
1074        stmdb   r8, {sp, lr}^           @ Save user sp, lr
1075        alignment_trap r4, r7, __temp_und
1076        zero_fp
1077        adrsvc  al, r9, ret_from_exception  @ r9  = normal FP return
1078        adrsvc  al, lr, fpundefinstr        @ lr  = undefined instr return
1079
1080    call_fpe:    get_current_task r10
1081        mov r8, #1
1082        strb    r8, [r10, #TSK_USED_MATH]   @ set current->used_math
1083        ldr r4, .LCfp
1084        add r10, r10, #TSS_FPESAVE      @ r10 = workspace
1085        ldr pc, [r4]            @ Call FP module USR entry point
1086
1087    fpundefinstr:    mrs r0, cpsr
1088        bic r0, r0, #MODE_MASK | I_BIT  @ preserve F-bit
1089        orr r0, r0, #MODE_SVC
1090        msr cpsr_c, r0          @ Enable interrupts
1091        mov r0, lr
1092        mov r1, sp
1093        adrsvc  al, lr, ret_from_exception
1094        b   SYMBOL_NAME(do_undefinstr)
1095
1096        .align  5
1097    __pabt_usr:  sub sp, sp, #S_FRAME_SIZE       @ Allocate frame size in one go
1098        stmia   sp, {r0 - r12}          @ Save r0 - r12
1099        ldr r4, .LCabt
1100        add r8, sp, #S_PC
1101        ldmia   r4, {r5 - r7}           @ Get USR pc, cpsr
1102        stmia   r8, {r5 - r7}           @ Save USR pc, cpsr, old_r0
1103        stmdb   r8, {sp, lr}^           @ Save sp_usr lr_usr
1104        alignment_trap r4, r7, __temp_abt
1105        zero_fp
1106        mrs r0, cpsr
1107        bic r0, r0, #MODE_MASK | I_BIT  @ preserve F-bit
1108        orr r0, r0, #MODE_SVC
1109        msr cpsr_c, r0          @ Enable interrupts
1110        mov r0, r5              @ address (pc)
1111        mov r1, sp              @ regs
1112        bl  SYMBOL_NAME(do_PrefetchAbort)   @ call abort handler
1113        /* fall through */
1114    /*
1115     * This is the return code to user mode for abort handlers
1116     */
1117    ENTRY(ret_from_exception)
1118        get_current_task tsk
1119        mov why, #0
1120        b   ret_to_user
```

```
1121
1122        .data
1123    ENTRY(fp_enter)
1124        .word   fpe_not_present
1125        .text
1126    /*
1127     * Register switch for ARMv3 and ARMv4 processors
1128     * r0 = previous, r1 = next, return previous.
1129     * previous and next are guaranteed not to be the same.
1130     */
1131    ENTRY(__switch_to)
1132        stmfd   sp!, {r4 - sl, fp, lr}      @ Store most regs on stack
1133        mrs ip, cpsr
1134        str ip, [sp, #-4]!          @ Save cpsr_SVC
1135        str sp, [r0, #TSS_SAVE]     @ Save sp_SVC
1136        ldr sp, [r1, #TSS_SAVE]     @ Get saved sp_SVC
1137    #ifndef CONFIG_UCLINUX
1138        ldr r2, [r1, #TSS_DOMAIN]
1139        mcr p15, 0, r2, c3, c0      @ Set domain register
1140    #endif
1141        ldr ip, [sp], #4
1142        msr spsr, ip            @ Save tasks CPSR into SPSR for this return
1143        ldmfd   sp!, {r4 - sl, fp, pc}^     @ Load all regs saved previously
1144
1145        .section ".text.init",#alloc,#execinstr
1146    /*
1147     * Vector stubs.  NOTE that we only align 'vector_IRQ' to a cache line boundary,
1148     * and we rely on each stub being exactly 48 (1.5 cache lines) in size.  This
1149     * means that we only ever load two cache lines for this code, or one if we're
1150     * lucky.  We also copy this code to 0x200 so that we can use branches in the
1151     * vectors, rather than ldr's.
1152     */
1153        .align  5
1154    __stubs_start:
1155    /*
1156     * Interrupt dispatcher
1157     * Enter in IRQ mode, spsr = SVC/USR CPSR, lr = SVC/USR PC
1158     */
1159    vector_IRQ:  @
1160        @ save mode specific registers
1161        @
1162        ldr r13, .LCsirq
1163        sub lr, lr, #4
1164        str lr, [r13]           @ save lr_IRQ
1165        mrs lr, spsr
1166        str lr, [r13, #4]           @ save spsr_IRQ
1167        @
```

```
1168        @ now branch to the relevent MODE handling routine
1169        @
1170        mrs r13, spsr           @ switch to SVC_32 mode
1171        bic r13, r13, #MODE_MASK        @ preserve F and T bits
1172        orr r13, r13, #MODE_SVC|I_BIT
1173        msr spsr_c, r13         @ switch to SVC_32 mode
1174
1175        and lr, lr, #15
1176        ldr lr, [pc, lr, lsl #2]
1177        movs    pc, lr              @ Changes mode and branches
1178
1179    .LCtab_irq:  .word   __irq_usr           @  0  (USR_26 / USR_32)
1180        .word   __irq_invalid           @  1  (FIQ_26 / FIQ_32)
1181        .word   __irq_invalid           @  2  (IRQ_26 / IRQ_32)
1182        .word   __irq_svc           @  3  (SVC_26 / SVC_32)
1183        .word   __irq_invalid           @  4
1184        .word   __irq_invalid           @  5
1185        .word   __irq_invalid           @  6
1186        .word   __irq_invalid           @  7
1187        .word   __irq_invalid           @  8
1188        .word   __irq_invalid           @  9
1189        .word   __irq_invalid           @  a
1190        .word   __irq_invalid           @  b
1191        .word   __irq_invalid           @  c
1192        .word   __irq_invalid           @  d
1193        .word   __irq_invalid           @  e
1194        .word   __irq_invalid           @  f
1195
1196        .align  5
1197
1198    /*
1199     * Data abort dispatcher - dispatches it to the correct handler for the processor
mode
1200     * Enter in ABT mode, spsr = USR CPSR, lr = USR PC
1201     */
1202    vector_data: @
1203        @ save mode specific registers
1204        @
1205        ldr r13, .LCsabt
1206        sub lr, lr, #8
1207        str lr, [r13]
1208        mrs lr, spsr
1209        str lr, [r13, #4]
1210        @
1211        @ now branch to the relevent MODE handling routine
1212        @
1213        mrs r13, spsr
```

```
1214        bic r13, r13, #MODE_MASK        @ preserve F and T bits
1215        orr r13, r13, #I_BIT | MODE_SVC
1216        msr spsr_c, r13         @ switch to SVC_32 mode
1217
1218        and lr, lr, #15
1219        ldr lr, [pc, lr, lsl #2]
1220        movs    pc, lr              @ Changes mode and branches
1221
1222    .LCtab_dabt: .word   __dabt_usr          @  0  (USR_26 / USR_32)
1223        .word   __dabt_invalid          @  1  (FIQ_26 / FIQ_32)
1224        .word   __dabt_invalid          @  2  (IRQ_26 / IRQ_32)
1225        .word   __dabt_svc          @  3  (SVC_26 / SVC_32)
1226        .word   __dabt_invalid          @  4
1227        .word   __dabt_invalid          @  5
1228        .word   __dabt_invalid          @  6
1229        .word   __dabt_invalid          @  7
1230        .word   __dabt_invalid          @  8
1231        .word   __dabt_invalid          @  9
1232        .word   __dabt_invalid          @  a
1233        .word   __dabt_invalid          @  b
1234        .word   __dabt_invalid          @  c
1235        .word   __dabt_invalid          @  d
1236        .word   __dabt_invalid          @  e
1237        .word   __dabt_invalid          @  f
1238
1239        .align  5
1240
1241    /*
1242     * Prefetch abort dispatcher - dispatches it to the correct handler for the
processor mode
1243     * Enter in ABT mode, spsr = USR CPSR, lr = USR PC
1244     */
1245    vector_prefetch:
1246        @
1247        @ save mode specific registers
1248        @
1249        ldr r13, .LCsabt
1250        sub lr, lr, #4
1251        str lr, [r13]           @ save lr_ABT
1252        mrs lr, spsr
1253        str lr, [r13, #4]           @ save spsr_ABT
1254        @
1255        @ now branch to the relevent MODE handling routine
1256        @
1257        mrs r13, spsr
1258        bic r13, r13, #MODE_MASK        @ preserve F and T bits
1259        orr r13, r13, #I_BIT | MODE_SVC
```

```
1260        msr spsr_c, r13         @ switch to SVC_32 mode
1261
1262        ands    lr, lr, #15
1263        ldr lr, [pc, lr, lsl #2]
1264        movs    pc, lr
1265
1266    .LCtab_pabt: .word   __pabt_usr          @  0 (USR_26 / USR_32)
1267        .word   __pabt_invalid          @  1 (FIQ_26 / FIQ_32)
1268        .word   __pabt_invalid          @  2 (IRQ_26 / IRQ_32)
1269        .word   __pabt_svc          @  3 (SVC_26 / SVC_32)
1270        .word   __pabt_invalid          @  4
1271        .word   __pabt_invalid          @  5
1272        .word   __pabt_invalid          @  6
1273        .word   __pabt_invalid          @  7
1274        .word   __pabt_invalid          @  8
1275        .word   __pabt_invalid          @  9
1276        .word   __pabt_invalid          @  a
1277        .word   __pabt_invalid          @  b
1278        .word   __pabt_invalid          @  c
1279        .word   __pabt_invalid          @  d
1280        .word   __pabt_invalid          @  e
1281        .word   __pabt_invalid          @  f
1282
1283        .align  5
1284
1285    /*
1286     * Undef instr entry dispatcher - dispatches it to the correct handler for the
processor mode
1287     * Enter in UND mode, spsr = SVC/USR CPSR, lr = SVC/USR PC
1288     */
1289    vector_undefinstr:
1290        @
1291        @ save mode specific registers
1292        @
1293        ldr r13, .LCsund
1294        str lr, [r13]           @ save lr_UND
1295        mrs lr, spsr
1296        str lr, [r13, #4]           @ save spsr_UND
1297        @
1298        @ now branch to the relevent MODE handling routine
1299        @
1300        mrs r13, spsr
1301        bic r13, r13, #MODE_MASK        @ preserve F and T bits
1302        orr r13, r13, #I_BIT | MODE_SVC
1303        msr spsr_c, r13         @ switch to SVC_32 mode
1304
1305        and lr, lr, #15
```

```
1306        ldr lr, [pc, lr, lsl #2]
1307        movs    pc, lr              @ Changes mode and branches
1308
1309    .LCtab_und:  .word   __und_usr           @  0 (USR_26 / USR_32)
1310        .word   __und_invalid           @  1 (FIQ_26 / FIQ_32)
1311        .word   __und_invalid           @  2 (IRQ_26 / IRQ_32)
1312        .word   __und_svc           @  3 (SVC_26 / SVC_32)
1313        .word   __und_invalid           @  4
1314        .word   __und_invalid           @  5
1315        .word   __und_invalid           @  6
1316        .word   __und_invalid           @  7
1317        .word   __und_invalid           @  8
1318        .word   __und_invalid           @  9
1319        .word   __und_invalid           @  a
1320        .word   __und_invalid           @  b
1321        .word   __und_invalid           @  c
1322        .word   __und_invalid           @  d
1323        .word   __und_invalid           @  e
1324        .word   __und_invalid           @  f
1325
1326        .align  5
1327
1328
/*=============================================================================
1329     * Undefined FIQs
1330
*-----------------------------------------------------------------------------
1331     * Enter in FIQ mode, spsr = ANY CPSR, lr = ANY PC
1332     * MUST PRESERVE SVC SPSR, but need to switch to SVC mode to show our msg.
1333     * Basically to switch modes, we *HAVE* to clobber one register...  brain
1334     * damage alert!  I don't think that we can execute any code in here in any
1335     * other mode than FIQ...  Ok you can switch to another mode, but you can't
1336     * get out of that mode without clobbering one register.
1337     */
1338    vector_FIQ:  disable_fiq
1339        subs    pc, lr, #4
1340
1341
/*=============================================================================
1342     * Address exception handler
1343
*-----------------------------------------------------------------------------
1344     * These aren't too critical.
1345     * (they're not supposed to happen, and won't happen in 32-bit data mode).
1346     */
1347
1348    vector_addrexcptn:
```

```
1349        b   vector_addrexcptn
1350
1351    /*
1352     * We group all the following data together to optimise
1353     * for CPUs with separate I & D caches.
1354     */
1355        .align  5
1356
1357    .LCvswi: .word   vector_swi
1358
1359    .LCsirq: .word   __temp_irq
1360    .LCsund: .word   __temp_und
1361    .LCsabt: .word   __temp_abt
1362
1363    __stubs_end:
1364
1365        .equ    __real_stubs_start, .LCvectors + 0x200
1366
1367    .LCvectors:  swi SYS_ERROR0
1368        b   __real_stubs_start + (vector_undefinstr - __stubs_start)
1369        ldr pc, __real_stubs_start + (.LCvswi - __stubs_start)
1370        b   __real_stubs_start + (vector_prefetch - __stubs_start)
1371        b   __real_stubs_start + (vector_data - __stubs_start)
1372        b   __real_stubs_start + (vector_addrexcptn - __stubs_start)
1373        b   __real_stubs_start + (vector_IRQ - __stubs_start)
1374        b   __real_stubs_start + (vector_FIQ - __stubs_start)
1375
1376    ENTRY(__trap_init)
1377        stmfd   sp!, {r4 - r6, lr}
1378
1379        mrs r1, cpsr        @ code from 2.0.38
1380        bic r1, r1, #MODE_MASK  @ clear mode bits
1381        orr r1, r1, #I_BIT|F_BIT|MODE_SVC   @ set SVC mode, disable IRQ,FIQ
1382        msr cpsr, r1
1383
1384        adr r1, .LCvectors          @ set up the vectors
1385        ldmia   r1, {r1, r2, r3, r4, r5, r6, ip, lr}
1386        stmia   r0, {r1, r2, r3, r4, r5, r6, ip, lr}
1387
1388        add r2, r0, #0x200
1389        adr r0, __stubs_start       @ copy stubs to 0x200
1390        adr r1, __stubs_end
1391    1:      ldr r3, [r0], #4
1392        str r3, [r2], #4
1393        cmp r0, r1
1394        blt 1b
1395        LOADREGS(fd, sp!, {r4 - r6, pc})
```

```
1396
1397        .data
1398
1399    /*
1400     * Do not reorder these, and do not insert extra data between...
1401     */
1402
1403    __temp_irq:  .word   0              @ saved lr_irq
1404         .word   0              @ saved spsr_irq
1405         .word   -1             @ old_r0
1406    __temp_und:  .word   0              @ Saved lr_und
1407         .word   0              @ Saved spsr_und
1408         .word   -1             @ old_r0
1409    __temp_abt:  .word   0              @ Saved lr_abt
1410         .word   0              @ Saved spsr_abt
1411         .word   -1             @ old_r0
1412
1413         .globl  SYMBOL_NAME(cr_alignment)
1414         .globl  SYMBOL_NAME(cr_no_alignment)
1415    SYMBOL_NAME(cr_alignment):
1416         .space  4
1417    SYMBOL_NAME(cr_no_alignment):
1418         .space  4
```

uClinux-dist/linux-2.4.x/arch/armnommu/kernel/irq.c

```
1   /*
2    *  linux/arch/arm/kernel/irq.c
3    *
4    *  Copyright (C) 1992 Linus Torvalds
5    *  Modifications for ARM processor Copyright (C) 1995-2000 Russell King.
6    *
7    * This program is free software; you can redistribute it and/or modify
8    * it under the terms of the GNU General Public License version 2 as
9    * published by the Free Software Foundation.
10    *
11    *  This file contains the code used by various IRQ handling routines:
12    *  asking for different IRQ's should be done through these routines
13    *  instead of just grabbing them. Thus setups with different IRQ numbers
14    *  shouldn't result in any weird surprises, and installing new handlers
15    *  should be easier.
16    *
17    *  IRQ's are in fact implemented a bit like signal handlers for the kernel.
18    *  Naturally it's not a 1:1 relation, but there are similarities.
19    */
20   #include <linux/config.h>
21   #include <linux/ptrace.h>
22   #include <linux/kernel_stat.h>
23   #include <linux/signal.h>
24   #include <linux/sched.h>
25   #include <linux/ioport.h>
26   #include <linux/interrupt.h>
27   #include <linux/slab.h>
28   #include <linux/random.h>
29   #include <linux/smp.h>
30   #include <linux/init.h>
31
32   #include <asm/irq.h>
33   #include <asm/system.h>
34   #include <asm/mach/irq.h>
35
36   #include <asm/arch/irq.h>  /* pick up fixup_irq definition */
37
38   /*
39    * Maximum IRQ count.  Currently, this is arbitary.  However, it should
40    * not be set too low to prevent false triggering.  Conversely, if it
41    * is set too high, then you could miss a stuck IRQ.
42    *
43    * Maybe we ought to set a timer and re-enable the IRQ at a later time?
44    */
45   #define MAX_IRQ_CNT    100000
```

```
46
47  static volatile unsigned long irq_err_count;
48  static spinlock_t irq_controller_lock;
49
50  struct irqdesc irq_desc[NR_IRQS];
51  void (*init_arch_irq)(void) __initdata = NULL;
52
53  /*
54   * Dummy mask/unmask handler
55   */
56  static void dummy_mask_unmask_irq(unsigned int irq)
57  {
58  }
59
60  /**
61   * disable_irq - disable an irq and wait for completion
62   * @irq: Interrupt to disable
63   *
64   * Disable the selected interrupt line.
65   *
66   * This function may be called - with care - from IRQ context.
67   */
68  void disable_irq(unsigned int irq)
69  {
70     unsigned long flags;
71
72     spin_lock_irqsave(&irq_controller_lock, flags);
73     irq_desc[irq].enabled = 0;
74     irq_desc[irq].mask(irq);
75     spin_unlock_irqrestore(&irq_controller_lock, flags);
76  }
77
78  /**
79   * enable_irq - enable interrupt handling on an irq
80   * @irq: Interrupt to enable
81   *
82   * Re-enables the processing of interrupts on this IRQ line
83   *
84   * This function may be called from IRQ context.
85   */
86  void enable_irq(unsigned int irq)
87  {
88     unsigned long flags;
89
90     spin_lock_irqsave(&irq_controller_lock, flags);
91     irq_desc[irq].probing = 0;
92     irq_desc[irq].triggered = 0;
```

```
93      irq_desc[irq].enabled = 1;
94      irq_desc[irq].unmask(irq);
95      spin_unlock_irqrestore(&irq_controller_lock, flags);
96    }
97
98    int get_irq_list(char *buf)
99    {
100     int i;
101     struct irqaction * action;
102     char *p = buf;
103
104     for (i = 0 ; i < NR_IRQS ; i++) {
105             action = irq_desc[i].action;
106         if (!action)
107             continue;
108         p += sprintf(p, "%3d: %10u ", i, kstat_irqs(i));
109         p += sprintf(p, "  %s", action->name);
110         for (action = action->next; action; action = action->next) {
111             p += sprintf(p, ", %s", action->name);
112         }
113         *p++ = '\n';
114     }
115
116   #ifdef CONFIG_ARCH_ACORN
117     p += get_fiq_list(p);
118   #endif
119     p += sprintf(p, "Err: %10lu\n", irq_err_count);
120     return p - buf;
121   }
122
123   /*
124    * IRQ lock detection.
125    *
126    * Hopefully, this should get us out of a few locked situations.
127    * However, it may take a while for this to happen, since we need
128    * a large number if IRQs to appear in the same jiffie with the
129    * same instruction pointer (or within 2 instructions).
130    */
131   static void check_irq_lock(struct irqdesc *desc, int irq, struct pt_regs *regs)
132   {
133     unsigned long instr_ptr = instruction_pointer(regs);
134
135     if (desc->lck_jif == jiffies &&
136         desc->lck_pc >= instr_ptr && desc->lck_pc < instr_ptr + 8) {
137         desc->lck_cnt += 1;
138
139         if (desc->lck_cnt > MAX_IRQ_CNT) {
```

```
140             printk(KERN_ERR "IRQ LOCK: IRQ%d is locking the system, disabled\n",
irq);
141             disable_irq(irq);
142         }
143     } else {
144         desc->lck_cnt = 0;
145         desc->lck_pc  = instruction_pointer(regs);
146         desc->lck_jif = jiffies;
147     }
148   }
149
150   /*
151    * do_IRQ handles all normal device IRQ's
152    */
153   asmlinkage void do_IRQ(int irq, struct pt_regs * regs)
154   {
155     struct irqdesc * desc;
156     struct irqaction * action;
157     int cpu;
158
159   #ifdef CONFIG_BOARD_SNDS100
160           /*
161            * FIXME: This is not the best place to put this work around.
162            */
163     CLEAR_PEND_INT(irq);
164   #endif
165
166   #ifdef CONFIG_ARCH_S3C44B0
167     CLEAR_PEND_INT(irq);
168   #endif
169
170     irq = fixup_irq(irq);
171
172     /*
173      * Some hardware gives randomly wrong interrupts.  Rather
174      * than crashing, do something sensible.
175      */
176     if (irq >= NR_IRQS)
177         goto bad_irq;
178
179     desc = irq_desc + irq;
180
181     spin_lock(&irq_controller_lock);
182     desc->mask_ack(irq);
183     spin_unlock(&irq_controller_lock);
184
185     cpu = smp_processor_id();
```

```
186     irq_enter(cpu, irq);
187     kstat.irqs[cpu][irq]++;
188     desc->triggered = 1;
189
190     /* Return with this interrupt masked if no action */
191     action = desc->action;
192
193     if (action) {
194         int status = 0;
195
196         if (desc->nomask) {
197             spin_lock(&irq_controller_lock);
198             desc->unmask(irq);
199             spin_unlock(&irq_controller_lock);
200         }
201
202         if (!(action->flags & SA_INTERRUPT))
203             __sti();
204
205         do {
206             status |= action->flags;
207             action->handler(irq, action->dev_id, regs);
208             action = action->next;
209         } while (action);
210
211         if (status & SA_SAMPLE_RANDOM)
212             add_interrupt_randomness(irq);
213         __cli();
214
215         if (!desc->nomask && desc->enabled) {
216             spin_lock(&irq_controller_lock);
217             desc->unmask(irq);
218             spin_unlock(&irq_controller_lock);
219         }
220     }
221
222     /*
223      * Debug measure - hopefully we can continue if an
224      * IRQ lockup problem occurs...
225      */
226     check_irq_lock(desc, irq, regs);
227
228     irq_exit(cpu, irq);
229
230     if (softirq_pending(cpu))
231         do_softirq();
232     return;
```

```
233
234   bad_irq:
235     irq_err_count += 1;
236     printk(KERN_ERR "IRQ: spurious interrupt %d\n", irq);
237     return;
238   }
239
240   #ifdef CONFIG_ARCH_ACORN
241   void do_ecard_IRQ(int irq, struct pt_regs *regs)
242   {
243     struct irqdesc * desc;
244     struct irqaction * action;
245     int cpu;
246
247     desc = irq_desc + irq;
248
249     cpu = smp_processor_id();
250     kstat.irqs[cpu][irq]++;
251     desc->triggered = 1;
252
253     action = desc->action;
254
255     if (action) {
256         do {
257             action->handler(irq, action->dev_id, regs);
258             action = action->next;
259         } while (action);
260     } else {
261         spin_lock(&irq_controller_lock);
262         desc->mask(irq);
263         spin_unlock(&irq_controller_lock);
264     }
265   }
266   #endif
267
268   int setup_arm_irq(int irq, struct irqaction * new)
269   {
270     int shared = 0;
271     struct irqaction *old, **p;
272     unsigned long flags;
273     struct irqdesc *desc;
274
275     /*
276      * Some drivers like serial.c use request_irq() heavily,
277      * so we have to be careful not to interfere with a
278      * running system.
279      */
```

```
280     if (new->flags & SA_SAMPLE_RANDOM) {
281         /*
282          * This function might sleep, we want to call it first,
283          * outside of the atomic block.
284          * Yes, this might clear the entropy pool if the wrong
285          * driver is attempted to be loaded, without actually
286          * installing a new handler, but is this really a problem,
287          * only the sysadmin is able to do this.
288          */
289             rand_initialize_irq(irq);
290     }
291 
292     /*
293      * The following block of code has to be executed atomically
294      */
295     desc = irq_desc + irq;
296     spin_lock_irqsave(&irq_controller_lock, flags);
297     p = &desc->action;
298     if ((old = *p) != NULL) {
299         /* Can't share interrupts unless both agree to */
300         if (!(old->flags & new->flags & SA_SHIRQ)) {
301             spin_unlock_irqrestore(&irq_controller_lock, flags);
302             return -EBUSY;
303         }
304 
305         /* add new interrupt at end of irq queue */
306         do {
307             p = &old->next;
308             old = *p;
309         } while (old);
310         shared = 1;
311     }
312 
313     *p = new;
314 
315     if (!shared) {
316         desc->nomask = (new->flags & SA_IRQNOMASK) ? 1 : 0;
317         desc->probing = 0;
318         if (!desc->noautoenable) {
319             desc->enabled = 1;
320             desc->unmask(irq);
321         }
322     }
323 
324     spin_unlock_irqrestore(&irq_controller_lock, flags);
325     return 0;
326 }
```

```
327
328  /**
329   *    request_irq - allocate an interrupt line
330   *    @irq: Interrupt line to allocate
331   *    @handler: Function to be called when the IRQ occurs
332   *    @irqflags: Interrupt type flags
333   *    @devname: An ascii name for the claiming device
334   *    @dev_id: A cookie passed back to the handler function
335   *
336   *    This call allocates interrupt resources and enables the
337   *    interrupt line and IRQ handling. From the point this
338   *    call is made your handler function may be invoked. Since
339   *    your handler function must clear any interrupt the board
340   *    raises, you must take care both to initialise your hardware
341   *    and to set up the interrupt handler in the right order.
342   *
343   *    Dev_id must be globally unique. Normally the address of the
344   *    device data structure is used as the cookie. Since the handler
345   *    receives this value it makes sense to use it.
346   *
347   *    If your interrupt is shared you must pass a non NULL dev_id
348   *    as this is required when freeing the interrupt.
349   *
350   *    Flags:
351   *
352   *    SA_SHIRQ        Interrupt is shared
353   *
354   *    SA_INTERRUPT        Disable local interrupts while processing
355   *
356   *    SA_SAMPLE_RANDOM    The interrupt can be used for entropy
357   *
358   */
359  int request_irq(unsigned int irq, void (*handler)(int, void *, struct pt_regs
*),
360          unsigned long irq_flags, const char * devname, void *dev_id)
361  {
362    unsigned long retval;
363    struct irqaction *action;
364
365    if (irq >= NR_IRQS || !irq_desc[irq].valid || !handler ||
366        (irq_flags & SA_SHIRQ && !dev_id))
367        return -EINVAL;
368
369    action = (struct irqaction *)kmalloc(sizeof(struct irqaction), GFP_KERNEL);
370    if (!action)
371        return -ENOMEM;
372
```

```
373     action->handler = handler;
374     action->flags = irq_flags;
375     action->mask = 0;
376     action->name = devname;
377     action->next = NULL;
378     action->dev_id = dev_id;
379
380     retval = setup_arm_irq(irq, action);
381
382     if (retval)
383         kfree(action);
384     return retval;
385   }
386
387   /**
388    *    free_irq - free an interrupt
389    *    @irq: Interrupt line to free
390    *    @dev_id: Device identity to free
391    *
392    *    Remove an interrupt handler. The handler is removed and if the
393    *    interrupt line is no longer in use by any driver it is disabled.
394    *    On a shared IRQ the caller must ensure the interrupt is disabled
395    *    on the card it drives before calling this function.
396    *
397    *    This function may be called from interrupt context.
398    */
399   void free_irq(unsigned int irq, void *dev_id)
400   {
401     struct irqaction * action, **p;
402     unsigned long flags;
403
404     if (irq >= NR_IRQS || !irq_desc[irq].valid) {
405         printk(KERN_ERR "Trying to free IRQ%d\n",irq);
406   #ifdef CONFIG_DEBUG_ERRORS
407         __backtrace();
408   #endif
409         return;
410     }
411
412     spin_lock_irqsave(&irq_controller_lock, flags);
413     for (p = &irq_desc[irq].action; (action = *p) != NULL; p = &action->next) {
414         if (action->dev_id != dev_id)
415             continue;
416
417             /* Found it - now free it */
418         *p = action->next;
419         kfree(action);
```

```
420         goto out;
421     }
422     printk(KERN_ERR "Trying to free free IRQ%d\n",irq);
423   #ifdef CONFIG_DEBUG_ERRORS
424     __backtrace();
425   #endif
426   out:
427     spin_unlock_irqrestore(&irq_controller_lock, flags);
428   }
429
430   /* Start the interrupt probing.  Unlike other architectures,
431    * we don't return a mask of interrupts from probe_irq_on,
432    * but return the number of interrupts enabled for the probe.
433    * The interrupts which have been enabled for probing is
434    * instead recorded in the irq_desc structure.
435    */
436   unsigned long probe_irq_on(void)
437   {
438     unsigned int i, irqs = 0;
439     unsigned long delay;
440
441     /*
442      * first snaffle up any unassigned but
443      * probe-able interrupts
444      */
445     spin_lock_irq(&irq_controller_lock);
446     for (i = 0; i < NR_IRQS; i++) {
447         if (!irq_desc[i].valid ||
448             !irq_desc[i].probe_ok ||
449             irq_desc[i].action)
450             continue;
451
452         irq_desc[i].probing = 1;
453         irq_desc[i].triggered = 0;
454         irq_desc[i].unmask(i);
455         irqs += 1;
456     }
457     spin_unlock_irq(&irq_controller_lock);
458
459     /*
460      * wait for spurious interrupts to mask themselves out again
461      */
462     for (delay = jiffies + HZ/10; time_before(jiffies, delay); )
463         /* min 100ms delay */;
464
465     /*
466      * now filter out any obviously spurious interrupts
```

```
467      */
468     spin_lock_irq(&irq_controller_lock);
469     for (i = 0; i < NR_IRQS; i++) {
470         if (irq_desc[i].probing &&
471             irq_desc[i].triggered) {
472             irq_desc[i].probing = 0;
473             irqs -= 1;
474         }
475     }
476     spin_unlock_irq(&irq_controller_lock);
477 
478     /* now filter out any obviously spurious interrupts */
479     return irqs;
480   }
481 
482   /*
483    * Possible return values:
484    *  >= 0 - interrupt number
485    *    -1 - no interrupt/many interrupts
486    */
487   int probe_irq_off(unsigned long irqs)
488   {
489     unsigned int i;
490     int irq_found = NO_IRQ;
491 
492     /*
493      * look at the interrupts, and find exactly one
494      * that we were probing has been triggered
495      */
496     spin_lock_irq(&irq_controller_lock);
497     for (i = 0; i < NR_IRQS; i++) {
498         if (irq_desc[i].probing &&
499             irq_desc[i].triggered) {
500             if (irq_found != NO_IRQ) {
501                 irq_found = NO_IRQ;
502                 goto out;
503             }
504             irq_found = i;
505         }
506     }
507 
508     if (irq_found == -1)
509         irq_found = NO_IRQ;
510   out:
511     spin_unlock_irq(&irq_controller_lock);
512 
513     return irq_found;
```

```
514   }
515
516   void __init init_irq_proc(void)
517   {
518   }
519
520   void __init init_IRQ(void)
521   {
522     extern void init_dma(void);
523     int irq;
524
525     for (irq = 0; irq < NR_IRQS; irq++) {
526         irq_desc[irq].probe_ok = 0;
527         irq_desc[irq].valid    = 0;
528         irq_desc[irq].noautoenable = 0;
529         irq_desc[irq].mask_ack = dummy_mask_unmask_irq;
530         irq_desc[irq].mask     = dummy_mask_unmask_irq;
531         irq_desc[irq].unmask   = dummy_mask_unmask_irq;
532     }
533
534     init_arch_irq();
535     init_dma();
536   }
```

uClinux-dist/linux-2.4.x/arch/armnommu/mm/proc-arm6,7.S

```
1   /*
2    *  linux/arch/armnommu/mm/proc-arm6,7.S
3    *
4    *  Copyright (C) 1997-2000 Russell King
5    *
6    * This program is free software; you can redistribute it and/or modify
7    * it under the terms of the GNU General Public License version 2 as
8    * published by the Free Software Foundation.
9    *
10    *  These are the low level assembler for performing cache and TLB
11    *  functions on the ARM610 & ARM710.
12    *
13    * Addition of S3C4530 by Arcturus Networks Inc.
14    *
15    */
16   #include <linux/linkage.h>
17   #include <asm/assembler.h>
18   #include <asm/constants.h>
19   #include <asm/procinfo.h>
20   #include <asm/errno.h>
21
22   /*
23    * Function: arm6_7_cache_clean_invalidate_all (void)
24    *    : arm6_7_cache_clean_invalidate_page (unsigned long address, int size, int
flags)
25    *
26    * Params  : address   Area start address
27    *    : size   size of area
28    *    : flags  b0 = I cache as well
29    *
30    * Purpose : Flush all cache lines
31    */
32   ENTRY(cpu_arm6_cache_clean_invalidate_all)
33   ENTRY(cpu_arm7_cache_clean_invalidate_all)
34   ENTRY(cpu_arm6_cache_clean_invalidate_range)
35   ENTRY(cpu_arm7_cache_clean_invalidate_range)
36   ENTRY(cpu_arm6_icache_invalidate_range)
37   ENTRY(cpu_arm7_icache_invalidate_range)
38   ENTRY(cpu_arm6_icache_invalidate_page)
39   ENTRY(cpu_arm7_icache_invalidate_page)
40   ENTRY(cpu_arm6_dcache_clean_range)
41   ENTRY(cpu_arm7_dcache_clean_range)
42   ENTRY(cpu_arm6_dcache_invalidate_range)
43   ENTRY(cpu_arm7_dcache_invalidate_range)
44           mov r0, #0
```

```
45  #ifdef CONFIG_CPU_WITH_CACHE
46  # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
47          mcr p15, 0, r0, c7, c0, 0       @ flush cache
48  # else
49  #  warning     "FIXME: Flush cache without MCR Instruction"
50  # endif
51  #endif
52  ENTRY(cpu_arm6_dcache_clean_page)
53  ENTRY(cpu_arm7_dcache_clean_page)
54  ENTRY(cpu_arm6_dcache_clean_entry)
55  ENTRY(cpu_arm7_dcache_clean_entry)
56  ENTRY(cpu_arm6_flush_ram_page)
57  ENTRY(cpu_arm7_flush_ram_page)
58          mov pc, lr
59
60  /*
61   * Function: arm6_7_tlb_invalidate_all (void)
62   *
63   * Purpose : flush all TLB entries in all caches
64   */
65  ENTRY(cpu_arm6_tlb_invalidate_all)
66  ENTRY(cpu_arm7_tlb_invalidate_all)
67          mov r0, #0
68  #ifndef NO_MM
69          mcr p15, 0, r0, c5, c0, 0       @ flush TLB
70  #endif
71          mov pc, lr
72
73  /*
74   * Function: arm6_7_tlb_invalidate_page (unsigned long address, int end, int
flags)
75   *
76   * Params  : address   Area start address
77   *         : end    Area end address
78   *         : flags  b0 = I cache as well
79   *
80   * Purpose : flush a TLB entry
81   */
82  ENTRY(cpu_arm6_tlb_invalidate_range)
83  ENTRY(cpu_arm7_tlb_invalidate_range)
84  #ifndef NO_MM
85  1:      mcr p15, 0, r0, c6, c0, 0       @ flush TLB
86          add r0, r0, #4096
87          cmp r0, r1
88          blt 1b
89  #endif
90          mov pc, lr
```

```
91
92  /*
93   * Function: arm6_7_tlb_invalidate_page (unsigned long address, int flags)
94   *
95   * Params  : address   Address
96   *    : flags  b0 = I-TLB as well
97   *
98   * Purpose : flush a TLB entry
99   */
100  ENTRY(cpu_arm6_tlb_invalidate_page)
101  ENTRY(cpu_arm7_tlb_invalidate_page)
102  #ifndef NO_MM
103        mcr p15, 0, r0, c6, c0, 0       @ flush TLB
104  #endif
105        mov pc, lr
106
107  /*
108   * Function: arm6_7_data_abort ()
109   *
110   * Params  : r0 = address of aborted instruction
111   *
112   * Purpose : obtain information about current aborted instruction
113   *
114   * Returns : r0 = address of abort
115   *       : r1 != 0 if writing
116   *       : r3 = FSR
117   *       : sp = pointer to registers
118   */
119
120  ENTRY(cpu_arm6_data_abort)
121        ldr r4, [r0]            @ read instruction causing problem
122        tst r4, r4, lsr #21         @ C = bit 20
123        sbc r1, r1, r1          @ r1 = C - 1
124        and r2, r4, #14 << 24
125        teq r2, #8 << 24            @ was it ldm/stm
126        bne Ldata_simple
127
128  Ldata_ldmstm: tst r4, #1 << 21            @ check writeback bit
129        beq Ldata_simple
130        mov r7, #0x11
131        orr r7, r7, r7, lsl #8
132        and r0, r4, r7
133        and r2, r4, r7, lsl #1
134        add r0, r0, r2, lsr #1
135        and r2, r4, r7, lsl #2
136        add r0, r0, r2, lsr #2
137        and r2, r4, r7, lsl #3
```

```
138         add r0, r0, r2, lsr #3
139         add r0, r0, r0, lsr #8
140         add r0, r0, r0, lsr #4
141         and r7, r0, #15         @ r7 = no. of registers to transfer.
142         and r5, r4, #15 << 16       @ Get Rn
143         ldr r0, [sp, r5, lsr #14]       @ Get register
144         tst r4, #1 << 23            @ U bit
145         subne   r7, r0, r7, lsl #2
146         addeq   r7, r0, r7, lsl #2      @ Do correction (signed)
147   Ldata_saver7: str r7, [sp, r5, lsr #14]       @ Put register
148   Ldata_simple:
149   #ifdef    NO_MM
150         orr r1, r2, #1          @ simulate FSR
151         mov r0, #0              @ gotta have something...
152   #else
153         mrc p15, 0, r0, c6, c0, 0       @ get FAR
154         mrc p15, 0, r3, c5, c0, 0       @ get FSR
155   #endif
156         and r3, r3, #255
157   #ifdef    NO_MM
158         mov r0, #1              @ return fail
159   #endif
160         mov pc, lr
161
162   ENTRY(cpu_arm7_data_abort)
163         ldr r4, [r0]            @ read instruction causing problem
164         tst r4, r4, lsr #21         @ C = bit 20
165         sbc r1, r1, r1          @ r1 = C - 1
166         and r2, r4, #15 << 24
167         add pc, pc, r2, lsr #22     @ Now branch to the relevent processing
routine
168         movs    pc, lr
169
170         b   Ldata_unknown
171         b   Ldata_unknown
172         b   Ldata_unknown
173         b   Ldata_unknown
174         b   Ldata_lateldrpostconst      @ ldr   rd, [rn], #m
175         b   Ldata_lateldrpreconst       @ ldr   rd, [rn, #m]    @ RegVal
176         b   Ldata_lateldrpostreg        @ ldr   rd, [rn], rm
177         b   Ldata_lateldrprereg     @ ldr   rd, [rn, rm]
178         b   Ldata_ldmstm            @ ldm*a rn, <rlist>
179         b   Ldata_ldmstm            @ ldm*b rn, <rlist>
180         b   Ldata_unknown
181         b   Ldata_unknown
182         b   Ldata_simple            @ ldc   rd, [rn], #m    @ Same as ldr  rd,
[rn], #m
```

```
183         b   Ldata_simple            @ ldc   rd, [rn, #m]
184         b   Ldata_unknown
185   Ldata_unknown:    @ Part of jumptable
186         mov r0, r2
187         mov r1, r4
188         mov r2, r3
189         bl  baddataabort
190   @FIXME        b   ret_from_sys_call
191
192   Ldata_lateldrpreconst:
193         tst r4, #1 << 21            @ check writeback bit
194         beq Ldata_simple
195   Ldata_lateldrpostconst:
196         movs    r2, r4, lsl #20         @ Get offset
197         beq Ldata_simple
198         and r5, r4, #15 << 16       @ Get Rn
199         ldr r0, [sp, r5, lsr #14]
200         tst r4, #1 << 23            @ U bit
201         subne   r7, r0, r2, lsr #20
202         addeq   r7, r0, r2, lsr #20
203         b   Ldata_saver7
204
205   Ldata_lateldrprereg:
206         tst r4, #1 << 21            @ check writeback bit
207         beq Ldata_simple
208   Ldata_lateldrpostreg:
209         and r5, r4, #15
210         ldr r2, [sp, r5, lsl #2]        @ Get Rm
211         mov r3, r4, lsr #7
212         ands    r3, r3, #31
213         and r6, r4, #0x70
214         orreq   r6, r6, #8
215         add pc, pc, r6
216         mov r0, r0
217
218         mov r2, r2, lsl r3          @ 0: LSL #!0
219         b   1f
220         b   1f              @ 1: LSL #0
221         mov r0, r0
222         b   1f              @ 2: MUL?
223         mov r0, r0
224         b   1f              @ 3: MUL?
225         mov r0, r0
226         mov r2, r2, lsr r3          @ 4: LSR #!0
227         b   1f
228         mov r2, r2, lsr #32         @ 5: LSR #32
229         b   1f
```

```
230         b   1f              @ 6: MUL?
231         mov r0, r0
232         b   1f              @ 7: MUL?
233         mov r0, r0
234         mov r2, r2, asr r3          @ 8: ASR #!0
235         b   1f
236         mov r2, r2, asr #32         @ 9: ASR #32
237         b   1f
238         b   1f              @ A: MUL?
239         mov r0, r0
240         b   1f              @ B: MUL?
241         mov r0, r0
242         mov r2, r2, ror r3          @ C: ROR #!0
243         b   1f
244         mov r2, r2, rrx         @ D: RRX
245         b   1f
246         mov r0, r0              @ E: MUL?
247         mov r0, r0
248         mov r0, r0              @ F: MUL?
249
250
251   1:        and r5, r4, #15 << 16       @ Get Rn
252         ldr r0, [sp, r5, lsr #14]
253         tst r4, #1 << 23            @ U bit
254         subne   r7, r0, r2
255         addeq   r7, r0, r2
256         b   Ldata_saver7
257
258   /*
259    * Function: arm6_7_check_bugs (void)
260    *       : arm6_7_proc_init (void)
261    *       : arm6_7_proc_fin (void)
262    *
263    * Notes   : This processor does not require these
264    */
265   ENTRY(cpu_arm6_check_bugs)
266   ENTRY(cpu_arm7_check_bugs)
267         mrs ip, cpsr
268         bic ip, ip, #F_BIT
269         msr cpsr, ip
270         mov pc, lr
271
272   ENTRY(cpu_arm6_proc_init)
273   ENTRY(cpu_arm7_proc_init)
274         mov pc, lr
275
276   ENTRY(cpu_arm6_proc_fin)
```

```
277   ENTRY(cpu_arm7_proc_fin)
278         mov r0, #F_BIT | I_BIT | SVC_MODE
279         msr cpsr_c, r0
280   #ifndef   NO_MM
281         mov r0, #0x31           @ ....S..DP...M
282         mcr p15, 0, r0, c1, c0, 0       @ disable caches
283   #else
284   # ifdef CONFIG_CPU_WITH_CACHE
285   #  ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
286         mov r0, #0x30           @ ........DP....
287         mcr p15, 0, r0, c1, c0, 0       @ disable caches
288   #  else
289   #   warning   "FIXME: Disable cache without MCR Instruction"
290   #  endif
291   # endif
292   #endif
293         mov pc, lr
294
295   ENTRY(cpu_arm6_do_idle)
296   ENTRY(cpu_arm7_do_idle)
297         mov r0, #-EINVAL
298         mov pc, lr
299
300   /*
301    * Function: arm6_7_set_pgd(unsigned long pgd_phys)
302    * Params  : pgd_phys Physical address of page table
303    * Purpose : Perform a task switch, saving the old processes state, and restoring
304    *         the new.
305    */
306   ENTRY(cpu_arm6_set_pgd)
307   ENTRY(cpu_arm7_set_pgd)
308         mov r1, #0
309   #ifdef CONFIG_CPU_WITH_CACHE
310   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
311         mcr p15, 0, r1, c7, c0, 0       @ flush cache
312   # else
313   #  warning        "FIXME: Flush cache without MCR Instruction"
314   # endif
315   #endif
316   #ifndef   NO_MM
317         mcr p15, 0, r0, c2, c0, 0       @ update page table ptr
318         mcr p15, 0, r1, c5, c0, 0       @ flush TLBs
319   #endif
320         mov pc, lr
321
322   /*
323    * Function: arm6_set_pmd ()
```

```
324     *
325     * Params  : r0 = Address to set
326     *         : r1 = value to set
327     *
328     * Purpose : Set a PMD and flush it out of any WB cache
329     */
330    ENTRY(cpu_arm6_set_pmd)
331          and r2, r1, #11
332          teq r2, #1
333          teqne   r2, #9
334          teqne   r2, #10
335          orreq   r1, r1, #16         @ Updatable = 1 if Page table/Cacheable
section
336          str r1, [r0]
337          mov pc, lr
338
339    /*
340     * Function: arm7_set_pmd ()
341     *
342     * Params  : r0 = Address to set
343     *         : r1 = value to set
344     *
345     * Purpose : Set a PMD and flush it out of any WB cache
346     */
347    ENTRY(cpu_arm7_set_pmd)
348          tst r1, #3
349          orrne   r1, r1, #16         @ Updatable bit is always set on ARM7
350          str r1, [r0]
351          mov pc, lr
352
353    /*
354     * Function: arm6_7_set_pte(pte_t *ptep, pte_t pte)
355     * Params  : r0 = Address to set
356     *         : r1 = value to set
357     * Purpose : Set a PTE and flush it out of any WB cache
358     */
359          .align  5
360    ENTRY(cpu_arm6_set_pte)
361    ENTRY(cpu_arm7_set_pte)
362          str r1, [r0], #-1024        @ linux version
363
364          eor r1, r1, #LPTE_PRESENT | LPTE_YOUNG | LPTE_WRITE | LPTE_DIRTY
365
366          bic r2, r1, #0xff0
367          bic r2, r2, #3
368          orr r2, r2, #HPTE_TYPE_SMALL
369
```

```
370         tst r1, #LPTE_USER | LPTE_EXEC  @ User or Exec?
371         orrne   r2, r2, #HPTE_AP_READ
372
373         tst r1, #LPTE_WRITE | LPTE_DIRTY    @ Write and Dirty?
374         orreq   r2, r2, #HPTE_AP_WRITE
375
376         tst r1, #LPTE_PRESENT | LPTE_YOUNG  @ Present and Young
377         movne   r2, #0
378
379         str r2, [r0]            @ hardware version
380         mov pc, lr
381
382   /*
383    * Function: _arm6_7_reset
384    * Params  : r0 = address to jump to
385    * Notes   : This sets up everything for a reset
386    */
387   ENTRY(cpu_arm6_reset)
388   ENTRY(cpu_arm7_reset)
389         mov r1, #0
390   #ifdef CONFIG_CPU_WITH_CACHE
391   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
392         mcr p15, 0, r1, c7, c0, 0       @ flush cache
393         mov r1, #0x30
394         mcr p15, 0, r1, c1, c0, 0       @ turn off Cache etc
395   # else
396   #  warning        "FIXME: Flush and Disable cache without MCR Instruction"
397   # endif
398   #endif
399   #ifndef   NO_MM
400         mcr p15, 0, r1, c5, c0, 0       @ flush TLB
401         mov r1, #0x30
402         mcr p15, 0, r1, c1, c0, 0       @ turn off MMU etc
403   #endif
404         mov pc, r0
405
406   cpu_armvlsi_name:
407         .asciz  "ARM/VLSI"
408   cpu_arm6_name:    .asciz  "ARM 6"
409   cpu_arm610_name:
410         .asciz  "ARM 610"
411   cpu_arm7_name:    .asciz  "ARM 7"
412   cpu_arm710_name:
413         .asciz  "ARM 710"
414   cpu_arm7tdmi_name:
415         .asciz  "ARM 7 TDMI"
416   cpu_at91_manu_name:
```

```
417         .asciz  "Atmel"
418   cpu_at91_name:
419         .asciz  "AT91M40xxx"
420   cpu_s3c3410_manu_name:
421   cpu_s3c4510b_manu_name:
422   cpu_s3c4530_manu_name:
423   cpu_s3c44b0_manu_name:
424         .asciz  "Samsung"
425   cpu_s3c3410_name:
426         .asciz  "S3C3410X"
427   cpu_s3c4510b_name:
428         .asciz  "S3C4510B"
429   cpu_s3c4530_name:
430         .asciz  "S3C4530A01"
431   cpu_s3c44b0_name:
432         .asciz  "S3C44B0"
433         .align
434
435         .section ".text.init", #alloc, #execinstr
436
437   __arm6_setup: mov r0, #F_BIT | I_BIT | SVC_MODE
438         msr cpsr_c, r0
439         mov r0, #0
440   #ifdef CONFIG_CPU_WITH_CACHE
441   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
442         mcr p15, 0, r0, c7, c0      @ flush caches on v3
443   # else
444   #  warning        "FIXME: Flush cache without MCR Instruction"
445         @ldr    r2, CACHE_CONTROL_MEM_LOCATION
446         @str    r0, [r2]
447   # endif
448   #endif
449   #ifndef NO_MM
450         mcr p15, 0, r0, c5, c0      @ flush TLBs on v3
451         mcr p15, 0, r4, c2, c0      @ load page table pointer
452         mov r0, #0x1f           @ Domains 0, 1 = client
453         mcr p15, 0, r0, c3, c0      @ load domain access register
454         mov r0, #0x3d           @ ....S..DPWC.M
455         orr r0, r0, #0x100
456   #endif
457   #ifdef CONFIG_CPU_WITH_CACHE
458   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
459         mov r0, #0x3c           @ .......DPWC..
460   # else
461   #  warning        "FIXME: Enable Cache and Others without MCR Instruction"
462   # endif
463   #else
```

```
464         @ Just in case r0 is modified by any code above
465         mov r0, #0x0
466   # warning     "FIXME: Setup anything if required"
467   #endif
468         mov pc, lr
469
470   __arm7_setup: mov r0, #F_BIT | I_BIT | SVC_MODE
471         msr cpsr_c, r0
472         mov r0, #0
473   #ifdef CONFIG_CPU_WITH_CACHE
474   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
475         mcr p15, 0, r0, c7, c0      @ flush caches on v3
476   # else
477   #  warning        "FIXME: Flush cache without MCR Instruction"
478         @ldr    r2, CACHE_CONTROL_MEM_LOCATION
479         @str    r0, [r2]
480   # endif
481   #endif
482   #ifndef   NO_MM
483         mcr p15, 0, r0, c5, c0      @ flush TLBs on v3
484         mcr p15, 0, r4, c2, c0      @ load page table pointer
485         mov r0, #0x1f           @ Domains 0, 1 = client
486         mcr p15, 0, r0, c3, c0      @ load domain access register
487         mov r0, #0x7d           @ ....S.LDPWC.M
488         orr r0, r0, #0x100
489   #endif
490   #ifdef CONFIG_CPU_WITH_CACHE
491   # ifdef CONFIG_CPU_WITH_MCR_INSTRUCTION
492         mov r0, #0x7c           @ ......LDPWC..
493   # else
494   #  warning        "FIXME: Enable Cache and Others without MCR Instruction"
495   # endif
496   #else
497         @ Just in case r0 is modified by any code above
498         mov r0, #0x0
499   # warning     "FIXME: Setup anything if required"
500   #endif
501         mov pc, lr
502
503   /*
504    * Purpose : Function pointers used to access above functions - all calls
505    *         come through these
506    */
507         .type   arm6_processor_functions, #object
508   ENTRY(arm6_processor_functions)
509         .word   cpu_arm6_data_abort
510         .word   cpu_arm6_check_bugs
```

```
511         .word   cpu_arm6_proc_init
512         .word   cpu_arm6_proc_fin
513         .word   cpu_arm6_reset
514         .word   cpu_arm6_do_idle
515
516         /* cache */
517         .word   cpu_arm6_cache_clean_invalidate_all
518         .word   cpu_arm6_cache_clean_invalidate_range
519         .word   cpu_arm6_flush_ram_page
520
521         /* dcache */
522         .word   cpu_arm6_dcache_invalidate_range
523         .word   cpu_arm6_dcache_clean_range
524         .word   cpu_arm6_dcache_clean_page
525         .word   cpu_arm6_dcache_clean_entry
526
527         /* icache */
528         .word   cpu_arm6_icache_invalidate_range
529         .word   cpu_arm6_icache_invalidate_page
530
531         /* tlb */
532         .word   cpu_arm6_tlb_invalidate_all
533         .word   cpu_arm6_tlb_invalidate_range
534         .word   cpu_arm6_tlb_invalidate_page
535
536         /* pgtable */
537         .word   cpu_arm6_set_pgd
538         .word   cpu_arm6_set_pmd
539         .word   cpu_arm6_set_pte
540         .size   arm6_processor_functions, . - arm6_processor_functions
541
542   /*
543    * Purpose : Function pointers used to access above functions - all calls
544    *         come through these
545    */
546         .type   arm7_processor_functions, #object
547   ENTRY(arm7_processor_functions)
548         .word   cpu_arm7_data_abort
549         .word   cpu_arm7_check_bugs
550         .word   cpu_arm7_proc_init
551         .word   cpu_arm7_proc_fin
552         .word   cpu_arm7_reset
553         .word   cpu_arm7_do_idle
554
555         /* cache */
556         .word   cpu_arm7_cache_clean_invalidate_all
557         .word   cpu_arm7_cache_clean_invalidate_range
```

```
558         .word   cpu_arm7_flush_ram_page
559
560         /* dcache */
561         .word   cpu_arm7_dcache_invalidate_range
562         .word   cpu_arm7_dcache_clean_range
563         .word   cpu_arm7_dcache_clean_page
564         .word   cpu_arm7_dcache_clean_entry
565
566         /* icache */
567         .word   cpu_arm7_icache_invalidate_range
568         .word   cpu_arm7_icache_invalidate_page
569
570         /* tlb */
571         .word   cpu_arm7_tlb_invalidate_all
572         .word   cpu_arm7_tlb_invalidate_range
573         .word   cpu_arm7_tlb_invalidate_page
574
575         /* pgtable */
576         .word   cpu_arm7_set_pgd
577         .word   cpu_arm7_set_pmd
578         .word   cpu_arm7_set_pte
579         .size   arm7_processor_functions, . - arm7_processor_functions
580
581         .type   cpu_arm6_info, #object
582   cpu_arm6_info:
583         .long   cpu_armvlsi_name
584         .long   cpu_arm6_name
585         .size   cpu_arm6_info, . - cpu_arm6_info
586
587         .type   cpu_arm610_info, #object
588   cpu_arm610_info:
589         .long   cpu_armvlsi_name
590         .long   cpu_arm610_name
591         .size   cpu_arm610_info, . - cpu_Arm610_info
592
593         .type   cpu_arm7_info, #object
594   cpu_arm7_info:
595         .long   cpu_armvlsi_name
596         .long   cpu_arm7_name
597         .size   cpu_arm7_info, . - cpu_arm7_info
598
599         .type   cpu_arm7tdmi_info, #object
600   cpu_arm7tdmi_info:
601         .long   cpu_armvlsi_name
602         .long   cpu_arm7tdmi_name
603         .size   cpu_arm7tdmi_info, . - cpu_arm7tdmi_info
604
```

```
605         .type   cpu_at91_info, #object
606   cpu_at91_info:
607         .long   cpu_at91_manu_name
608         .long   cpu_at91_name
609         .size   cpu_at91_info, . - cpu_at91_info
610
611         .type   cpu_s3c3410_info, #object
612   cpu_s3c3410_info:
613         .long   cpu_s3c3410_manu_name
614         .long   cpu_s3c3410_name
615         .size   cpu_s3c3410_info, . - cpu_s3c3410b_info
616
617         .type   cpu_s3c44b0_info, #object
618   cpu_s3c44b0_info:
619         .long   cpu_s3c44b0_manu_name
620         .long   cpu_s3c44b0_name
621         .size   cpu_s3c44b0_info, . - cpu_s3c44b0_info
622         .type   cpu_s3c4510b_info, #object
623
624   cpu_s3c4510b_info:
625         .long   cpu_s3c4510b_manu_name
626         .long   cpu_s3c4510b_name
627         .size   cpu_s3c4510b_info, . - cpu_s3c4510b_info
628
629                 .type   cpu_s3c4530_info, #object
630   cpu_s3c4530_info:
631                 .long   cpu_s3c4530_manu_name
632                 .long   cpu_s3c4530_name
633                 .size   cpu_s3c4530_info, . - cpu_s3c4530_info
634
635         .type   cpu_arm710_info, #object
636   cpu_arm710_info:
637         .long   cpu_armvlsi_name
638         .long   cpu_arm710_name
639         .size   cpu_arm710_info, . - cpu_arm710_info
640
641         .type   cpu_arch_name, #object
642   cpu_arch_name:    .asciz  "armv3"
643         .size   cpu_arch_name, . - cpu_arch_name
644
645         .type   cpu_elf_name, #object
646   cpu_elf_name: .asciz  "v3"
647         .size   cpu_elf_name, . - cpu_elf_name
648
649         .align
650         .section ".proc.info", #alloc, #execinstr
651
```

```
652         .type   __arm6_proc_info, #object
653   __arm6_proc_info:
654         .long   0x41560600
655         .long   0xffffff0
656         .long   0x00000c1e
657         b   __arm6_setup
658         .long   cpu_arch_name
659         .long   cpu_elf_name
660         .long   HWCAP_SWP | HWCAP_26BIT
661         .long   cpu_arm6_info
662         .long   arm6_processor_functions
663         .size   __arm6_proc_info, . - __arm6_proc_info
664
665         .type   __arm610_proc_info, #object
666   __arm610_proc_info:
667         .long   0x41560610
668         .long   0xffffff0
669         .long   0x00000c1e
670         b   __arm6_setup
671         .long   cpu_arch_name
672         .long   cpu_elf_name
673         .long   HWCAP_SWP | HWCAP_26BIT
674         .long   cpu_arm610_info
675         .long   arm6_processor_functions
676         .size   __arm610_proc_info, . - __arm610_proc_info
677
678         .type   __arm7_proc_info, #object
679   __arm7_proc_info:
680         .long   0x41007000
681         .long   0xffffff00
682         .long   0x00000c1e
683         b   __arm7_setup
684         .long   cpu_arch_name
685         .long   cpu_elf_name
686         .long   HWCAP_SWP | HWCAP_26BIT
687         .long   cpu_arm7_info
688         .long   arm7_processor_functions
689         .size   __arm7_proc_info, . - __arm7_proc_info
690
691         .type   __arm710_proc_info, #object
692   __arm710_proc_info:
693         .long   0x41007100
694         .long   0xfff8ff00
695         .long   0x00000c1e
696         b   __arm7_setup
697         .long   cpu_arch_name
698         .long   cpu_elf_name
```

```
699         .long   HWCAP_SWP | HWCAP_26BIT
700         .long   cpu_arm710_info
701         .long   arm7_processor_functions
702         .size   __arm710_proc_info, . - __arm710_proc_info
703
704         .type   __arm7tdmi_proc_info, #object
705   __arm7tdmi_proc_info:
706         .long   0x41007700
707         .long   0xfff8ff00
708         .long   0x00000c1e
709         b   __arm7_setup
710         .long   cpu_arch_name
711         .long   cpu_elf_name
712         .long   HWCAP_SWP | HWCAP_26BIT
713         .long   cpu_arm7tdmi_info
714         .long   arm7_processor_functions
715         .size   __arm7tdmi_proc_info, . - __arm7tdmi_proc_info
716
717         .type   __at91_proc_info, #object
718   __at91_proc_info:
719         .long   0x14000040
720         .long   0xfff000e0
721         .long   0x00000c1e
722         b   __arm7_setup
723         .long   cpu_arch_name
724         .long   cpu_elf_name
725         .long   HWCAP_SWP | HWCAP_26BIT
726         .long   cpu_at91_info
727         .long   arm7_processor_functions
728         .size   __at91_proc_info, . - __at91_proc_info
729
730         .type   __s3c4510b_proc_info, #object
731   __s3c4510b_proc_info:
732                 .long   0x36365000                      @ cpu_val
733                 .long   0xffffff000                     @ cpu_mask
734                 .long   0x00000c1e                      @ __cpu_mmu_flags
735                 b       __arm7_setup                    @ __cpu_flush
736                 .long   cpu_arch_name                   @ arch_name
737                 .long   cpu_elf_name                    @ elf_name
738                 .long   HWCAP_SWP | HWCAP_26BIT         @ elf_hwcap
739                 .long   cpu_s3c4510b_info               @ info
740                 .long   arm7_processor_functions        @ info
741                 .size   __s3c4510b_proc_info, . - __s3c4510b_proc_info
742
743         .type   __s3c44b0_proc_info, #object
744   __s3c44b0_proc_info:
745         .long   0x36366000
```

```
746         .long   0xFFFFF000
747         .long   0x00000c1e
748         b   __arm7_setup
749         .long   cpu_arch_name
750         .long   cpu_elf_name
751         .long   HWCAP_SWP | HWCAP_26BIT
752         .long   cpu_s3c44b0_info
753         .long   arm7_processor_functions
754         .size   __s3c44b0_proc_info, . - __s3c44b0_proc_info
755
756         .type   __s3c4530_proc_info, #object
757   __s3c4530_proc_info:
758         .long   0x4c000000          @ cpu_val
759         .long   0xfff000e0          @ cpu_mask
760         .long   0x00000c1e          @ __cpu_mmu_flags
761         b   __arm7_setup            @ __cpu_flush
762         .long   cpu_arch_name           @ arch_name
763         .long   cpu_elf_name            @ elf_name
764         .long   HWCAP_SWP | HWCAP_26BIT     @ elf_hwcap
765         .long   cpu_s3c4530_info        @ info
766         .long   arm7_processor_functions    @ info
767         .size   __s3c4530_proc_info, . - __s3c4530_proc_info
768
769         .type   __s3c3410_proc_info, #object
770   __s3c3410_proc_info:
771         .long   0x34103410          @ cpu_val
772         .long   0xffff0000          @ cpu_mask
773         .long   0x00000c1e          @ __cpu_mmu_flags
774         b   __arm7_setup            @ __cpu_flush
775         .long   cpu_arch_name           @ arch_name
776         .long   cpu_elf_name            @ elf_name
777         .long   HWCAP_SWP | HWCAP_26BIT     @ elf_hwcap
778         .long   cpu_s3c3410_info        @ info
779         .long   arm7_processor_functions    @ info
780         .size   __s3c3410_proc_info, . - __s3c3410_proc_info
781
```

uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/arch.c

```
1   /*
2    *  linux/arch/arm/mach-s3c44b0/arch.c
3    *
4    *  Architecture specific fixups.  This is where any
5    *  parameters in the params struct are fixed up, or
6    *  any additional architecture specific information
7    *  is pulled from the params struct.
8    */
9   #include <linux/tty.h>
10   #include <linux/delay.h>
11   #include <linux/pm.h>
12   #include <linux/init.h>
13
14   #include <asm/elf.h>
15   #include <asm/setup.h>
16   #include <asm/mach-types.h>
17   #include <asm/mach/arch.h>
18
19   extern void genarch_init_irq(void);
20
21   MACHINE_START(S3C44B0, "51EDA")
22           MAINTAINER("Lei Ni")
23           BOOT_MEM(DRAM_BASE,0x00000000,0x00000000)
24           INITIRQ(genarch_init_irq)
25   MACHINE_END
```

uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/irq.c

```
1   /*
2   *  linux/arch/armnommu/mach-s3c44b0/irq.c
3   *  2001 Mac Wang <mac@os.nctu.edu.tw>
4   */
5   #include <linux/init.h>
6
7   #include <asm/mach/irq.h>
8   #include <asm/hardware.h>
9   #include <asm/io.h>
10   #include <asm/irq.h>
11   #include <asm/system.h>
12
13   void s3c44b0_mask_irq(unsigned int irq)
14   {
15      INT_DISABLE(irq);
16   }
17
18   void s3c44b0_unmask_irq(unsigned int irq)
19   {
20      INT_ENABLE(irq);
21   }
22
23   void s3c44b0_mask_ack_irq(unsigned int irq)
24   {
25      INT_DISABLE(irq);
26   }
27
28   void s3c44b0_int_init()
29   {
30      IntPend = 0x1FFFFF;
31      IntMode = INT_MODE_IRQ;
32      INT_ENABLE(INT_GLOBAL);
33      *(unsigned int *)INTCON = 5;
34   }
```

uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/time.c

```
1   /*
2    * time.c  Timer functions for Samsung 44B0
3    */
4
5   #include <linux/time.h>
6   #include <linux/timex.h>
7   #include <linux/types.h>
8   #include <linux/sched.h>
9   #include <asm/io.h>
10   #include <asm/arch/hardware.h>
11
12   unsigned long samsung_gettimeoffset (void)
13   {
14      return 0;
15   }
16
17   void samsung_timer_interrupt(int irq, void *dev_id, struct pt_regs *regs)
18   {
19      do_timer(regs);
20   }
```

uClinux-dist/linux-2.4.x/arch/armnommu/mach-s3c44b0/Makefile

```
1   #
2   # Makefile for the linux kernel.
3   #
4   # Note! Dependencies are done automagically by 'make dep', which also
5   # removes any old dependencies. DON'T put your own dependencies here
6   # unless it's something special (ie not a .c file).
7
8   USE_STANDARD_AS_RULE := true
9
10   O_TARGET       := s3c44b0.o
11
12   # Object file lists.
13
14   obj-y          := $(patsubst %.c, %.o, $(wildcard *.c))
15   obj-m          :=
16   obj-n          :=
17   obj-           :=
18
19   export-objs        :=
20
21   include $(TOPDIR)/Rules.make
```

uClinux-dist/linux-2.4.x/arch/armnommu/tools/mach-types

```
1   # Database of machine macros and numbers
2   #
3   # Please do not send patches to this file; it is automatically generated!
4   # To add an entry into this database, please see Documentation/arm/README,
5   # or contact rmk@arm.linux.org.uk
6   #
7   # Last update: Mon Jun 11 15:20:04 2001
8   #
9   # machine_is_xxx    CONFIG_xxxx     MACH_TYPE_xxx       number
10   #
11   ebsa110            ARCH_EBSA110        EBSA110         0
12   riscpc         ARCH_RPC        RISCPC          1
13   nexuspci       ARCH_NEXUSPCI       NEXUSPCI        3
14   ebsa285            ARCH_EBSA285        EBSA285         4
15   netwinder      ARCH_NETWINDER      NETWINDER       5
16   cats           ARCH_CATS       CATS            6
17   tbox           ARCH_TBOX       TBOX            7
18   co285          ARCH_CO285      CO285           8
19   clps7110       ARCH_CLPS7110       CLPS7110        9
20   archimedes     ARCH_ARC        ARCHIMEDES      10
21   a5k            ARCH_A5K        A5K         11
22   etoile         ARCH_ETOILE     ETOILE          12
23   lacie_nas      ARCH_LACIE_NAS      LACIE_NAS       13
24   clps7500       ARCH_CLPS7500       CLPS7500        14
25   shark          ARCH_SHARK      SHARK           15
26   brutus         SA1100_BRUTUS       BRUTUS          16
27   personal_server        ARCH_PERSONAL_SERVER    PERSONAL_SERVER     17
28   itsy           SA1100_ITSY     ITSY            18
29   l7200          ARCH_L7200      L7200           19
30   pleb           SA1100_PLEB     PLEB            20
31   integrator     ARCH_INTEGRATOR     INTEGRATOR      21
32   bitsy          SA1100_BITSY        BITSY           22
33   ixp1200            ARCH_IXP1200        IXP1200         23
34   p720t          ARCH_P720T      P720T           24
35   assabet            SA1100_ASSABET      ASSABET         25
36   victor         SA1100_VICTOR       VICTOR          26
37   lart           SA1100_LART     LART            27
38   ranger         SA1100_RANGER       RANGER          28
39   graphicsclient     SA1100_GRAPHICSCLIENT   GRAPHICSCLIENT      29
40   xp860          SA1100_XP860        XP860           30
41   cerf           SA1100_CERF     CERF            31
42   nanoengine     SA1100_NANOENGINE   NANOENGINE      32
43   fpic           SA1100_FPIC     FPIC            33
44   extenex1       SA1100_EXTENEX1     EXTENEX1        34
45   sherman            SA1100_SHERMAN      SHERMAN         35
```

```
46   accelent_sa         SA1100_ACCELENT      ACCELENT_SA      36
47   accelent_l7200      ARCH_L7200_ACCELENT  ACCELENT_L7200       37
48   netport             SA1100_NETPORT       NETPORT          38
49   pangolin        SA1100_PANGOLIN      PANGOLIN         39
50   yopy            SA1100_YOPY      YOPY             40
51   coolidge        SA1100_COOLIDGE      COOLIDGE         41
52   huw_webpanel        SA1100_HUW_WEBPANEL  HUW_WEBPANEL         42
53   spotme          ARCH_SPOTME      SPOTME           43
54   freebird        ARCH_FREEBIRD        FREEBIRD         44
55   ti925           ARCH_TI925       TI925            45
56   riscstation         ARCH_RISCSTATION     RISCSTATION      46
57   cavy            SA1100_CAVY      CAVY             47
58   jornada720      SA1100_JORNADA720    JORNADA720       48
59   omnimeter       SA1100_OMNIMETER     OMNIMETER        49
60   edb7211             ARCH_EDB7211         EDB7211          50
61   citygo          SA1100_CITYGO        CITYGO           51
62   pfs168          SA1100_PFS168        PFS168           52
63   spot            SA1100_SPOT      SPOT             53
64   flexanet        SA1100_FLEXANET      FLEXANET         54
65   webpal          ARCH_WEBPAL      WEBPAL           55
66   linpda          SA1100_LINPDA        LINPDA           56
67   anakin          ARCH_ANAKIN      ANAKIN           57
68   mvi             SA1100_MVI       MVI          58
69   jupiter             SA1100_JUPITER       JUPITER          59
70   psionw          ARCH_PSIONW      PSIONW           60
71   aln             SA1100_ALN       ALN          61
72   camelot             ARCH_CAMELOT         CAMELOT          62
73   gds2200             SA1100_GDS2200       GDS2200          63
74   psion_series7       SA1100_PSION_SERIES7     PSION_SERIES7        64
75   xfile           SA1100_XFILE         XFILE            65
76   accelent_ep9312         ARCH_ACCELENT_EP9312     ACCELENT_EP9312      66
77   ic200           ARCH_IC200       IC200            67
78   creditlart      SA1100_CREDITLART    CREDITLART       68
79   htm             SA1100_HTM       HTM          69
80   iq80310             ARCH_IQ80310         IQ80310          70
81   freebot             SA1100_FREEBOT       FREEBOT          71
82   entel           ARCH_ENTEL       ENTEL            72
83   enp3510             ARCH_ENP3510         ENP3510          73
84   trizeps             SA1100_TRIZEPS       TRIZEPS          74
85   nesa            SA1100_NESA      NESA             75
86   venus           ARCH_VENUS       VENUS            76
87   tardis          ARCH_TARDIS      TARDIS           77
88   mercury             ARCH_MERCURY         MERCURY          78
89   empeg           SA1100_EMPEG         EMPEG            79
90   i80200fcc       ARCH_I80200FCC       I80200FCC        80
91   itt_cpb             SA1100_ITT_CPB       ITT_CPB          81
92   sa1110_svc      ARCH_SA1110_SVC      SA1110_SVC       82
```

```
93   sa1100          SA1100_SA1100       SA1100          83
94   alpha2          SA1100_ALPHA2       ALPHA2          84
95   alpha1          SA1100_ALPHA1       ALPHA1          85
96   netarm          ARCH_NETARM     NETARM          86
97
98   # The following are unallocated
99
100   p52            ARCH_P52        P52         87
101   spipe          ARCH_SPIPE      SPIPE           88
102   atmel          ARCH_ATMEL      ATMEL           89
103   dsc21                   ARCH_DSC21              DSC21                   115
104   snds100            ARCH_SNDS100        SNDS100         90
105   s3c44b0            ARCH_S3C44B0        S3C44B0         178
106   evS3C4530HEI       BOARD_EVS3C4530HEI  EVS3C4530HEI        164
107   S3C3410X       BOARD_SMDK40100     S3C3410         165
108
109
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/dma.h

```
1   /*
2    * asm/arch-armnommu/arch-p52/dma.h:
3    * Mindspeed 2001
4    */
5
6   #ifndef __ASM_ARCH_DMA_H
7   #define __ASM_ARCH_DMA_H
8
9
10   #define MAX_DMA_ADDRESS         0x01000000
11   #define MAX_DMA_CHANNELS         13
12   #define arch_dma_init(dma_chan)
13
14   #endif /* _ASM_ARCH_DMA_H */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/irq.h

```
1   /*
2    * asm/arch-samsung/irq.h:
3    * 2001 Mac Wang <mac@os.nctu.edu.tw>
4    */
5
6   #ifndef __ASM_ARCH_IRQ_H__
7   #define __ASM_ARCH_IRQ_H__
8
9   #include <asm/hardware.h>
10   #include <asm/io.h>
11   #include <asm/mach/irq.h>
12   #include <asm/arch/irqs.h>
13
14   #define fixup_irq(x) (x)
15
16   extern void s3c44b0_mask_irq(unsigned int irq);
17   extern void s3c44b0_unmask_irq(unsigned int irq);
18   extern void s3c44b0_mask_ack_irq(unsigned int irq);
19   extern void s3c44b0_int_init(void);
20
21   static __inline__ void irq_init_irq(void)
22   {
23      unsigned long flags;
24      int irq;
25
26
27      save_flags_cli(flags);
28      s3c44b0_int_init();
29      restore_flags(flags);
30
31      for (irq = 0; irq < NR_IRQS; irq++) {
32          irq_desc[irq].valid = 1;
33          irq_desc[irq].probe_ok  = 1;
34          irq_desc[irq].mask_ack  = s3c44b0_mask_ack_irq;
35          irq_desc[irq].mask  = s3c44b0_mask_irq;
36          irq_desc[irq].unmask    = s3c44b0_unmask_irq;
37      }
38   }
39   #endif /* __ASM_ARCH_IRQ_H__ */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/param.h

```
1  #ifndef __ARCH_ASM_PARAM_H__
2  #define __ARCH_ASM_PARAM_H__
3
4  /* nothing for now */
5
6  #endif /* __ARCH_ASM_PARAM_H__ */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/timex.h

```
1  /*
2   * timex.h:
3   * 2001 Mac Wang <mac@os.nctu.edu.tw>
4   */
5  #ifndef __ASM_ARCH_TIMEX_H__
6  #define __ASM_ARCH_TIMEX_H__
7
8  #include <asm/hardware.h>
9
10  //#define CLOCK_TICK_RATE     (((fMCLK_MHz/100))*2)
11  #define    S3C44B0_TIMER_FREQ  100 // 100Hz
12
13  #endif /* __ASM_ARCH_TIMEX_H__ */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/vmalloc.h

```
1   /*
2    *  linux/include/asm-arm/arch-dsc21/vmalloc.h
3    *
4    *  Copyright (C) 2000 Russell King.
5    *
6    * This program is free software; you can redistribute it and/or modify
7    * it under the terms of the GNU General Public License as published by
8    * the Free Software Foundation; either version 2 of the License, or
9    * (at your option) any later version.
10    *
11    * This program is distributed in the hope that it will be useful,
12    * but WITHOUT ANY WARRANTY; without even the implied warranty of
13    * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14    * GNU General Public License for more details.
15    *
16    * You should have received a copy of the GNU General Public License
17    * along with this program; if not, write to the Free Software
18    * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
19    */
20
21   /*
22    * Just any arbitrary offset to the start of the vmalloc VM area: the
23    * current 8MB value just means that there will be a 8MB "hole" after the
24    * physical memory until the kernel virtual memory starts.  That means that
25    * any out-of-bounds memory accesses will hopefully be caught.
26    * The vmalloc() routines leaves a hole of 4kB between each vmalloced
27    * area for the same reason. ;)
28    */
29   #define VMALLOC_OFFSET   (8*1024*1024)
30   #define VMALLOC_START    (((unsigned  long)high_memory  +  VMALLOC_OFFSET)  &
~(VMALLOC_OFFSET-1))
31   #define VMALLOC_VMADDR(x) ((unsigned long)(x))
32   #define VMALLOC_END       (PAGE_OFFSET + 0x10000000)
33
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/hardware.h

```
1   /***********************************************************/
2   /*                            */
3   /* Samsung KS32C4510b                    */
4   /* Mac Wang <mac@os.nctu.edu.tw>            */
5   /*                            */
6   /***********************************************************/
7   #ifndef __ASM_ARCH_HARDWARE_H
8   #define __ASM_ARCH_HARDWARE_H
9
10   /*
11    * define S3C4510b CPU master clock
12    */
13   #define MHz        1000000
14   #define fMCLK_MHz  (60 * MHz)
15
16
17   /*
18    * ASIC Address Definition
19    */
20
21   #define Base_Addr  (0x1c00000)//(0x3ff0000)
22
23   #define VPint  *(volatile unsigned int *)
24   #define VPshort    *(volatile unsigned short *)
25   #define VPchar *(volatile unsigned char *)
26
27   #ifndef CSR_WRITE
28   #   define CSR_WRITE(addr,data)     (VPint(addr) = (data))
29   #endif
30
31   #ifndef CSR_READ
32   #   define CSR_READ(addr)  (VPint(addr))
33   #endif
34
35
36
37   /* ************************ */
38   /* System Manager Registers */
39   /* ************************ */
40   #define SYSCFG     0x01c00000
41
42
43   /* ************************ */
44   /* Ethernet BDMA Registers */
45   /* ************************ */
```

```
46
47  /*
48   * CAM      0x9100 ~ 0x917C
49   * BDMATXBUF   0x9200 ~ 0x92FC
50   * BDMARXBUF   0x9800 ~ 0x99FC
51   */
52
53  /* ********************** */
54  /* Ethernet MAC Registers */
55  /* ********************** */
56
57
58  /* ************************ */
59  /* HDLC Channel A Registers */
60  /* ************************ */
61
62  /* ************************ */
63  /* HDLC Channel B Registers */
64  /* ************************ */
65
66  /* ******************** */
67  /* I/O Ports Registers */
68  /* ******************** */
69
70
71  /* ****************************** */
72  /* Interrupt Controller Registers */
73  /* ****************************** */
74  #define    INTCON      (Base_Addr+0x200000)
75  #define INTMOD     (Base_Addr+0x200008)//(Base_Addr+0x4000)
76  #define INTPND     (Base_Addr+0x200020)//(Base_Addr+0x4004)    this is I_ISPR
77  #define    INTCLEAR    (Base_Addr+0x200024)// I add, this I_ISPC
78  #define INTMSK     (Base_Addr+0x20000c)//(Base_Addr+0x4008)
79
80  #define    IntConR     (VPint(INTCON))
81  #define IntMode        (VPint(INTMOD))
82  #define IntPend        (VPint(INTPND))
83  #define    IntClear    (VPint(INTCLEAR))   // I add
84  #define IntMask        (VPint(INTMSK))
85
86
87  #define INT_MODE_IRQ   0x000000
88  #define INT_MODE_FIQ   0x3ffffff//0x1FFFFF
89  #define INT_MASK_DIS   0x3ffffff//0x1FFFFF
90  #define INT_MASK_ENA   0x000000
91
92  #define INT_ENABLE(n)      IntMask &= ~(1<<(n))
```

```
93   #define INT_DISABLE(n)      IntMask |= (1<<(n))
94   #define CLEAR_PEND_INT(n)  IntClear = (1<<(n))//IntPend = (1<<(n))
95   #define SET_PEND_INT(n)        //IntPndTst |= (1<<(n))
96
97   /* ****************** */
98   /* I2C Bus Registers */
99   /* ****************** */
100
101   /* *************** */
102   /* GDMA Registers */
103   /* *************** */
104
105   /* *************** */
106   /* UART Registers */
107   /* *************** */
108
109   #define DEBUG_CONSOLE (0)
110
111   #define ULCON0         (Base_Addr+0x100000)//(Base_Addr+0xD000)
112   #define ULCON1         (Base_Addr+0x104000)//(Base_Addr+0xE000)
113   #define UCON0      (Base_Addr+0x100004)//(Base_Addr+0xD004)
114   #define UCON1      (Base_Addr+0x104004)//(Base_Addr+0xE004)
115   #define USTAT0         (Base_Addr+0x100010)//(Base_Addr+0xD008)
116   #define USTAT1         (Base_Addr+0x104010)//(Base_Addr+0xE008)
117   #define UTXBUF0        (Base_Addr+0x100020)//(Base_Addr+0xD00C)
118   #define UTXBUF1        (Base_Addr+0x104020)//(Base_Addr+0xE00C)
119   #define URXBUF0        (Base_Addr+0x100024)//(Base_Addr+0xD010)
120   #define URXBUF1        (Base_Addr+0x104024)//(Base_Addr+0xE010)
121   #define UBRDIV0        (Base_Addr+0x100028)//(Base_Addr+0xD014)
122   #define UBRDIV1        (Base_Addr+0x104028)//(Base_Addr+0xE014)
123
124   #define UART_BASE0    ULCON0
125   #define UART_BASE1    ULCON1
126
127   #if DEBUG_CONSOLE == 0
128     #define DEBUG_TX_BUFF_BASE  UTXBUF0
129     #define DEBUG_RX_BUFF_BASE  URXBUF0
130     #define DEBUG_UARTLCON_BASE ULCON0
131     #define DEBUG_UARTCONT_BASE UCON0
132     #define DEBUG_UARTBRD_BASE  UBRDIV0
133     #define DEBUG_CHK_STAT_BASE USTAT0
134   #elif DEBUG_CONSOLE == 1
135     #define DEBUG_TX_BUFF_BASE  UTXBUF1
136     #define DEBUG_RX_BUFF_BASE  URXBUF1
137     #define DEBUG_UARTLCON_BASE ULCON1
138     #define DEBUG_UARTCONT_BASE UCON1
139     #define DEBUG_UARTBRD_BASE  UBRDIV1
```

```
140     #define DEBUG_CHK_STAT_BASE USTAT1
141   #endif
142
143   #define DEBUG_ULCON_REG_VAL   (0x3)
144   #define DEBUG_UCON_REG_VAL    (0x45)//(0x9)
145   #define DEBUG_UBRDIV_REG_VAL  ((unsigned
int)((fMCLK_MHz)/(57600*16)+0.5)-1)//(53)//(0x500)
146   #define DEBUG_RX_CHECK_BIT    (1)//(0X20)
147   #define DEBUG_TX_CAN_CHECK_BIT    (2)//(0X40)
148   #define DEBUG_TX_DONE_CHECK_BIT   (4)//(0X80)
149
150
151   /* ***************** */
152   /* Timers Registers */
153   /* ***************** */
154
155
156   #define rTCFG0        (*(volatile unsigned int *)0x1d50000)
157   #define rTCFG1        (*(volatile unsigned int *)0x1d50004)
158   #define rTCON     (*(volatile unsigned int *)0x1d50008)
159
160   #define rTCNTB0       (*(volatile unsigned int *)0x1d5000c)
161   #define rTCMPB0       (*(volatile unsigned int *)0x1d50010)
162   #define rTCNTO0       (*(volatile unsigned int *)0x1d50014)
163
164   #define rTCNTB1       (*(volatile unsigned int *)0x1d50018)
165   #define rTCMPB1       (*(volatile unsigned int *)0x1d5001c)
166   #define rTCNTO1       (*(volatile unsigned int *)0x1d50020)
167
168   #define rTCNTB2       (*(volatile unsigned int *)0x1d50024)
169   #define rTCMPB2       (*(volatile unsigned int *)0x1d50028)
170   #define rTCNTO2       (*(volatile unsigned int *)0x1d5002c)
171
172   #define rTCNTB3       (*(volatile unsigned int *)0x1d50030)
173   #define rTCMPB3       (*(volatile unsigned int *)0x1d50034)
174   #define rTCNTO3       (*(volatile unsigned int *)0x1d50038)
175
176   #define rTCNTB4       (*(volatile unsigned int *)0x1d5003c)
177   #define rTCMPB4       (*(volatile unsigned int *)0x1d50040)
178   #define rTCNTO4       (*(volatile unsigned int *)0x1d50044)
179
180   #define rTCNTB5       (*(volatile unsigned int *)0x1d50048)
181   #define rTCNTO5       (*(volatile unsigned int *)0x1d5004c)
182
183   /********************/
184   /* SYSCFG Register */
185   /********************/
```

```
186
187
188   /***********************************/
189   /* System Memory Control Register */
190   /***********************************/
191
192   /*****************************************/
193   /* ROMCON0: ROM Bank 0 Control Register */
194   /*****************************************/
195
196
197   /*********************************************/
198   /* SDRAMCON0: SDRAM Bank 0 Control Register */
199   /*********************************************/
200
201
202   /*************************************************/
203   /* DRAM Refresh & External I/O Control Register */
204   /*************************************************/
205
206   /********/
207   /* Misc */
208   /********/
209
210   #define HARD_RESET_NOW()
211
212
213   #endif /* __ASM_ARCH_HARDWARE_H */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/hardware.h

```
1   /*
2    * asm/arch-samsung/irqs.h:
3    * Mac Wang <mac@os.nctu.edu.tw>
4    */
5   #ifndef __ASM_ARCH_IRQS_H__
6   #define __ASM_ARCH_IRQS_H__
7   #define NR_IRQS     26//21
8   #define VALID_IRQ(i)    (i<=8 ||(i>=16 && i<NR_IRQS))
9   /*
10   #define INT_EXTINT0    0
11   #define INT_EXTINT1    1
12   #define INT_EXTINT2    2
13   #define INT_EXTINT3    3
14   #define INT_UARTTX0    4
15   #define INT_UARTRX0    5
16   #define INT_UARTTX1    6
17   #define INT_UARTRX1    7
18   #define INT_GDMA0  8
19   #define INT_GDMA1  9
20   #define INT_TIMER0 10
21   #define INT_TIMER1 11
22   #define INT_HDLCTXA    12
23   #define INT_HDLCRXA    13
24   #define INT_HDLCTXB    14
25   #define INT_HDLCRXB    15
26   #define INT_BDMATX 16
27   #define INT_BDMARX 17
28   #define INT_MACTX  18
29   #define INT_MACRX  19
30   #define INT_IIC        20
31   #define INT_GLOBAL 21
32
33   #define IRQ_TIMER  INT_TIMER0
34   */
35
36   #define INT_GLOBAL 26
37   #define INT_EXTINT0    25
38   #define INT_EXTINT1    24
39   #define INT_EXTINT2    23
40   #define INT_EXTINT3    22
41   #define    INT_EXTINT4_7   21
42   #define    INT_TICK    20
43   #define    INT_ZDMA0   19
44   #define    INT_ZDMA1   18
45   #define    INT_BDMA0   17
46   #define    INT_BDMA1   16
```

```
47  #define    INT_WDT     15
48  #define    INT_UERR0_1 14
49  #define    INT_TIMER0  13
50  #define    INT_TIMER1  12
51  #define    INT_TIMER2  11
52  #define    INT_TIMER3  10
53  #define    INT_TIMER4  9
54  #define    INT_TIMER5  8
55  #define INT_UARTRX0    7
56  #define INT_UARTRX1    6
57  #define INT_IIC        5
58  #define INT_SIO        4
59  #define INT_UARTTX0    3
60  #define INT_UARTTX1    2
61  #define INT_RTC        1
62  #define INT_ADC        0
63  #define IRQ_TIMER  INT_TIMER5
64
65  #endif /* __ASM_ARCH_IRQS_H__ */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/hardware.h

```
1   /*
2    * uclinux/include/asm-armnommu/arch-atmel/mmu.h
3    *
4    */
5   #ifndef __ASM_ARCH_PROCESSOR_H
6   #define __ASM_ARCH_PROCESSOR_H
7
8   #define EISA_bus 0
9   #define EISA_bus__is_a_macro
10   #define MCA_bus 0
11   #define MCA_bus__is_a_macro
12
13   #define TASK_SIZE (0x01a00000UL)
14
15   #endif
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/system.h

```
1   /*
2    * linux/include/asm-armnommu/arch-dsc21/system.h
3    *
4    * Copyright (c) 1999 Nicolas Pitre <nico@cam.org>
5    * Copyright (c) 2001 RidgeRun, Inc (http://www.ridgerun.org)
6    *
7    */
8
9   static inline void arch_idle(void)
10   {
11      while (!current->need_resched && !hlt_counter);
12   }
13
14   extern inline void arch_reset(char mode)
15   {
16      /* REVISIT --gmcnutt */
17   }
18
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/uncompress.c

```
1   /*
2    * linux/include/asm/arch-samsung/uncompress.c
3    * 2001 Mac Wang <mac@os.nctu.edu.tw>
4    */
5
6   #include <asm/hardware.h>
7
8   static int s3c44b0_decomp_setup()
9   {
10      CSR_WRITE(DEBUG_UARTLCON_BASE, DEBUG_ULCON_REG_VAL);
11      CSR_WRITE(DEBUG_UARTCONT_BASE, DEBUG_UCON_REG_VAL);
12      CSR_WRITE(DEBUG_UARTBRD_BASE,  DEBUG_UBRDIV_REG_VAL);
13   }
14
15   static int s3c44b0_putc(char c)
16   {
17      CSR_WRITE(DEBUG_TX_BUFF_BASE, c);
18      while(!(CSR_READ(DEBUG_CHK_STAT_BASE) & DEBUG_TX_DONE_CHECK_BIT));
19
20      if(c == '\n')
21          s3c44b0_putc('\r');
22   }
23
24   static void s3c44b0_puts(const char *s)
25   {
26      while(*s != '\0')
27          s3c44b0_putc(*s++);
28   }
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/uncompress.h

```
1   /*
2    * asm/arch/uncompress.c:
3    *         Optional routines to aid in debugging the decompression phase
4    *         of kernel boot.
5    * copyright:
6    *         (C) 2001 RidgeRun, Inc. (http://www.ridgerun.com)
7    * author: Gordon McNutt <gmcnutt@ridgerun.com>
8    *
9    * This program is free software; you can redistribute  it and/or modify it
10    * under  the terms of  the GNU General  Public License as published by the
11    * Free Software Foundation;  either version 2 of the  License, or (at your
12    * option) any later version.
13    *
14    * THIS  SOFTWARE  IS PROVIDED   ``AS  IS'' AND   ANY  EXPRESS OR IMPLIED
15    * WARRANTIES,   INCLUDING, BUT NOT  LIMITED  TO, THE IMPLIED WARRANTIES OF
16    * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN
17    * NO  EVENT  SHALL   THE AUTHOR  BE    LIABLE FOR ANY   DIRECT, INDIRECT,
18    * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
19    * NOT LIMITED   TO, PROCUREMENT OF  SUBSTITUTE GOODS  OR SERVICES; LOSS OF
20    * USE, DATA,  OR PROFITS; OR  BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
21    * ANY THEORY OF LIABILITY, WHETHER IN  CONTRACT, STRICT LIABILITY, OR TORT
22    * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
23    * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
24    *
25    * You should have received a copy of the  GNU General Public License along
26    * with this program; if not, write  to the Free Software Foundation, Inc.,
27    * 675 Mass Ave, Cambridge, MA 02139, USA.
28    *
29    */
30   #include <asm/arch/uncompress.c>
31   /*
32    * This is used by arch/armnommu/boot/compressed/misc.c to write progress info
33    * out the serial port so that the user can see debug messages up to the point
34    * where the kernel is decompressed. The STANDALONE_DEBUG macro chooses between
35    * this and the standard printf. Punt.
36    * --gmcnutt
37    */
38   #define puts(s)             s3c44b0_puts(s)
39
40   /*
41    * Not sure what this is for. Probably an optional watchdog to check if the
42    * decompress got hung so we can warn the user. Punt.
43    */
44   #define arch_decomp_wdog()
45
```

```
46  /*
47   * If we need to do some setup prior to decompression (like initializing the
48   * UART if we want to use puts() above) then we define it here. Punt.
49   */
50  #define arch_decomp_setup()    s3c44b0_decomp_setup()
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/io.h

```
1   /*
2    * linux/include/asm-armnommu/arch-dsc21/io.h
3    *
4    * Copyright (C) 1997-1999 Russell King
5    *
6    * Modifications:
7    *  06-12-1997  RMK Created.
8    *  07-04-1999  RMK Major cleanup
9    *  02-19-2001  gjm     Leveraged for armnommu/dsc21
10    */
11   #ifndef __ASM_ARM_ARCH_IO_H
12   #define __ASM_ARM_ARCH_IO_H
13
14   /*
15    * kernel/resource.c uses this to initialize the global ioport_resource struct
16    * which is used in all calls to request_resource(), allocate_resource(), etc.
17    * --gmcnutt
18    */
19   #define IO_SPACE_LIMIT 0xffffffff
20
21   /*
22    * If we define __io then asm/io.h will take care of most of the inb & friends
23    * macros. It still leaves us some 16bit macros to deal with ourselves, though.
24    * We don't have PCI or ISA on the dsc21 so I dropped __mem_pci & __mem_isa.
25    * --gmcnutt
26    */
27   #define PCIO_BASE 0
28   #define __io(a) (PCIO_BASE + (a))
29   #define __arch_getw(a) (*(volatile unsigned short *)(a))
30   #define __arch_putw(v,a) (*(volatile unsigned short *)(a) = (v))
31
32   /*
33    * Defining these two gives us ioremap for free. See asm/io.h.
34    * --gmcnutt
35    */
36   #define iomem_valid_addr(iomem,sz) (1)
37   #define iomem_to_phys(iomem) (iomem)
38
39   #endif
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/memory.h

```
1   /*
2    * linux/include/asm-armnommu/arch-p52/memory.h
3    *
4    * Copyright (c) 1999 Nicolas Pitre <nico@cam.org>
5    * 2001 Mindspeed
6    */
7
8   #ifndef __ASM_ARCH_MEMORY_H
9   #define __ASM_ARCH_MEMORY_H
10
11   #define __virt_to_bus(x) ((unsigned long) (x))
12   #define __bus_to_virt(x) ((void *) (x))
13   #define __virt_to_phys(x) ((unsigned long) (x))
14   #define __phys_to_virt(x) ((void *) (x))
15
16   #define TASK_SIZE  (0x01a00000UL)
17   #define TASK_SIZE_26   TASK_SIZE
18
19   #define PHYS_OFFSET    (DRAM_BASE)
20   #define PAGE_OFFSET PHYS_OFFSET
21   #define END_MEM     (DRAM_BASE + DRAM_SIZE)
22   #endif
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/serial.h

```
1   /*
2    * linux/include/asm/arch-samsung/serial.h
3    * 2001 Mac Wang <mac@os.nctu.edu.tw>
4    */
5   #ifndef __ASM_ARCH_SERIAL_H
6   #define __ASM_ARCH_SERIAL_H
7
8   #include <asm/arch/hardware.h>
9   #include <asm/irq.h>
10
11   #define RS_TABLE_SIZE  2
12   #define BASE_BAUD  9600
13   #define STD_COM_FLAGS  (ASYNC_BOOT_AUTOCONF | ASYNC_SKIP_TEST)
14   #define STD_SERIAL_PORT_DEFNS      \
15      /* UART CLK PORT IRQ FLAGS */   \
16      { 0, BASE_BAUD, UART_BASE0, INT_UARTRX0, STD_COM_FLAGS },   /* ttyS0 */ \
17      { 0, BASE_BAUD, UART_BASE1, INT_UARTRX1, STD_COM_FLAGS },   /* ttyS1 */
18   #define EXTRA_SERIAL_PORT_DEFNS
19
20   #define UART_LCR   0x00        /* Line Control Register    */
21   #define UART_IER   0x04        /* Global Control Register  */
22   #define UART_GCR   0x04        /* Global Control Register  */
23   #define UART_LSR   0x08        /* Line Status Reigster     */
24   #define UART_TX        0x0C        /* TX buffer Register       */
25   #define UART_RX        0x10        /* RX buffer Register       */
26   #define UART_BDR   0x14        /* Baudrate Divisor Reigster    */
27
28   #define UART_LSR_OE    0x01        // Overrun error
29   #define UART_LSR_PE    0x02        // Parity error
30   #define UART_LSR_FE    0x04        // Frame error
31   #define UART_LSR_BI    0x08        // Break detect
32   #define UART_LSR_DTR   0x10        // Data terminal ready
33   #define UART_LSR_DR    0x20        // Receive data ready
34   #define UART_LSR_THRE  0x40        // Transmit buffer register empty
35   #define UART_LSR_TEMT  0x80        // Transmit complete
36
37   #define UART_LCR_WLEN5 0x00
38   #define UART_LCR_WLEN6 0x01
39   #define UART_LCR_WLEN7 0x02
40   #define UART_LCR_WLEN8 0x03
41   #define UART_LCR_PARITY    0x20
42   #define UART_LCR_NPAR  0x00
43   #define UART_LCR_OPAR  0x20
44   #define UART_LCR_EPAR  0x28
45   #define UART_LCR_SPAR  0x10
```

```
46  #define UART_LCR_SBC   0x40
47
48  #define UART_GCR_RX_INT    0x01
49  #define UART_GCR_TX_INT    0x08
50  #define UART_GCR_RX_STAT_INT   0x04
51
52  #define UART_IER_MSI
53  #define UART_IER_RLSI  0x04
54  #define UART_IER_THRI  0x08
55  #define UART_IER_RDI   0x01
56
57  #define UART_MSR       0
58  #define UART_MSR_DCD       0
59  #define UART_MSR_RI        0
60  #define UART_MSR_DSR       0
61  #define UART_MSR_CTS       0
62  #define UART_MSR_DDCD      0
63  #define UART_MSR_TERI      0
64  #define UART_MSR_DDSR      0
65  #define UART_MSR_DCTS      0
66  #define UART_MSR_ANY_DELTA 0
67
68  #define PORT_S3C4510B  14
69
70  struct serial_baudtable
71  {
72     unsigned int baudrate;
73     unsigned int div;
74  };
75
76  struct serial_baudtable uart_baudrate[] =
77  {
78     {  1200, 0x5150},
79     {  2400, 0x28A0},
80     {  4800, 0x1440},
81     {  9600, 0x0A20},
82     { 19200, 0x0500},
83     { 38400, 0x0280},
84     { 57600, 0x01A0},
85     {115200, 0x00D0},
86     {230400, 0x0060},
87     {460860, 0x0020}
88  };
89
90  unsigned int baudrate_div(unsigned int baudrate)
91  {
92     int i;
```

```
93      int len = sizeof(uart_baudrate)/sizeof(struct serial_baudtable);
94      for(i = 0; i < len; i++)
95          if(uart_baudrate[i].baudrate == baudrate)
96              return uart_baudrate[i].div;
97      return 0;
98  }
99
100 #define disable_uart_tx_interrupt(line)       \
101 {                       \
102   if(line) {              \
103       INT_DISABLE(INT_UARTTX1);   \
104       CLEAR_PEND_INT(INT_UARTTX1);    \
105   }                       \
106   else {                  \
107       INT_DISABLE(INT_UARTTX0);   \
108       CLEAR_PEND_INT(INT_UARTTX0);    \
109   }                       \
110 }
111
112 #define disable_uart_rx_interrupt(line)       \
113 {                       \
114   if(line) {              \
115       INT_DISABLE(INT_UARTRX1);   \
116       CLEAR_PEND_INT(INT_UARTRX1);    \
117   }                       \
118   else {                  \
119       INT_DISABLE(INT_UARTRX0);   \
120       CLEAR_PEND_INT(INT_UARTRX0);    \
121   }                       \
122 }
123
124 #define enable_uart_tx_interrupt(line)        \
125 {                       \
126   if(line) {              \
127       INT_ENABLE(INT_UARTTX1);    \
128       SET_PEND_INT(INT_UARTTX1);  \
129   }                       \
130   else {                  \
131       INT_ENABLE(INT_UARTTX0);    \
132       SET_PEND_INT(INT_UARTTX0);  \
133   }                       \
134 }
135
136 #define enable_uart_rx_interrupt(line)        \
137 {                       \
138   if(line) {              \
139       INT_ENABLE(INT_UARTRX1);    \
```

```
140     }                   \
141     else {                  \
142         INT_ENABLE(INT_UARTRX0);    \
143     }                   \
144   }
145
146   #endif /* __ASM_ARCH_SERIAL_H */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/arch-s3c44b0/time.h

```
1   /*
2    * linux/include/asm/arch-samsung/time.h
3    * 2001 Mac Wang <mac@os.nctu.edu.tw>
4    */
5
6   #ifndef __ASM_ARCH_TIME_H__
7   #define __ASM_ARCH_TIME_H__
8
9   #include <asm/uaccess.h>
10   #include <asm/io.h>
11   #include <asm/hardware.h>
12   #include <asm/arch/timex.h>
13
14   extern struct irqaction timer_irq;
15
16   extern unsigned long samsung_gettimeoffset(void);
17   extern void samsung_timer_interrupt(int irq, void *dev_id, struct pt_regs *regs);
18
19
20   void __inline__ setup_timer (void)
21   {
22      rTCON   &= 0xf0ffffff;
23      rTCFG0  &= 0xff00ffff;
24      rTCFG0  |= (16-1)<<16;
25      rTCFG1  &= 0xfff0fffff;
26      rTCFG1  |= 0<<20;
27      rTCNTB5  = fMCLK_MHz/(S3C44B0_TIMER_FREQ*16*2);
28      rTCON   |= 0x02000000;
29      rTCON   &= 0xf0ffffff;
30      rTCON   |= 0x05000000;
31      CLEAR_PEND_INT(IRQ_TIMER);
32      INT_ENABLE(IRQ_TIMER);
33
34      timer_irq.handler = samsung_timer_interrupt;
35      setup_arm_irq(IRQ_TIMER, &timer_irq);
36   }
37
38   #endif /* __ASM_ARCH_TIME_H__ */
```

uClinux-dist/linux-2.4.x/include/asm-armnommu/proc-armv/system.h

```
1   /*
2    *  linux/include/asm-arm/proc-armv/system.h
3    *
4    *  Copyright (C) 1996 Russell King
5    *
6    * This program is free software; you can redistribute it and/or modify
7    * it under the terms of the GNU General Public License version 2 as
8    * published by the Free Software Foundation.
9    */
10  #ifndef __ASM_PROC_SYSTEM_H
11  #define __ASM_PROC_SYSTEM_H
12
13  #include <linux/config.h>
14
15  #define set_cr(x)                   \
16     __asm__ __volatile__(               \
17     "mcr    p15, 0, %0, c1, c0  @ set CR"   \
18     : : "r" (x))
19
20  #define CR_M   (1 << 0)    /* MMU enable               */
21  #define CR_A   (1 << 1)    /* Alignment abort enable       */
22  #define CR_C   (1 << 2)    /* Dcache enable            */
23  #define CR_W   (1 << 3)    /* Write buffer enable          */
24  #define CR_P   (1 << 4)    /* 32-bit exception handler     */
25  #define CR_D   (1 << 5)    /* 32-bit data address range        */
26  #define CR_L   (1 << 6)    /* Implementation defined       */
27  #define CD_B   (1 << 7)    /* Big endian               */
28  #define CR_S   (1 << 8)    /* System MMU protection        */
29  #define CD_R   (1 << 9)    /* ROM MMU protection           */
30  #define CR_F   (1 << 10)   /* Implementation defined       */
31  #define CR_Z   (1 << 11)   /* Implementation defined       */
32  #define CR_I   (1 << 12)   /* Icache enable            */
33  #define CR_V   (1 << 13)   /* Vectors relocated to 0xffff0000  */
34  #define CR_RR  (1 << 14)   /* Round Robin cache replacement    */
35
36  extern unsigned long cr_no_alignment;  /* defined in entry-armv.S */
37  extern unsigned long cr_alignment; /* defined in entry-armv.S */
38
39  #ifdef CONFIG_ARCH_S3C44B0
40  #define vectors_base() (0x0C000000)
41  #elif
42  #define    vectors_base()   (0)
43  #endif
44
45  /*
```

```
46   * A couple of speedups for the ARM
47   */
48
49  /*
50   * Save the current interrupt enable state & disable IRQs
51   */
52  #define __save_flags_cli(x)                    \
53     ({                           \
54         unsigned long temp;             \
55     __asm__ __volatile__(                   \
56     "mrs    %0, cpsr        @ save_flags_cli\n" \
57  "  orr %1, %0, #128\n"                 \
58  "  msr cpsr_c, %1"                \
59     : "=r" (x), "=r" (temp)                 \
60     :                            \
61     : "memory");                         \
62     })
63
64  /*
65   * Enable IRQs
66   */
67  #define __sti()                             \
68     ({                           \
69         unsigned long temp;             \
70     __asm__ __volatile__(                   \
71     "mrs    %0, cpsr        @ sti\n"        \
72  "  bic %0, %0, #128\n"                 \
73  "  msr cpsr_c, %0"                \
74     : "=r" (temp)                        \
75     :                            \
76     : "memory");                         \
77     })
78
79  /*
80   * Disable IRQs
81   */
82  #define __cli()                             \
83     ({                           \
84         unsigned long temp;             \
85     __asm__ __volatile__(                   \
86     "mrs    %0, cpsr        @ cli\n"        \
87  "  orr %0, %0, #128\n"                 \
88  "  msr cpsr_c, %0"                \
89     : "=r" (temp)                        \
90     :                            \
91     : "memory");                         \
92     })
```

```
93
94  /*
95   * Enable FIQs
96   */
97  #define __stf()                             \
98    ({                            \
99        unsigned long temp;             \
100   __asm__ __volatile__(                   \
101   "mrs    %0, cpsr        @ stf\n"        \
102  " bic %0, %0, #64\n"                    \
103  " msr cpsr_c, %0"                 \
104   : "=r" (temp)                         \
105   :                             \
106   : "memory");                          \
107   })
108
109 /*
110  * Disable FIQs
111  */
112 #define __clf()                             \
113   ({                            \
114       unsigned long temp;             \
115   __asm__ __volatile__(                   \
116   "mrs    %0, cpsr        @ clf\n"        \
117  " orr %0, %0, #64\n"                    \
118  " msr cpsr_c, %0"                 \
119   : "=r" (temp)                         \
120   :                             \
121   : "memory");                          \
122   })
123
124 /*
125  * save current IRQ & FIQ state
126  */
127 #define __save_flags(x)                       \
128   __asm__ __volatile__(                   \
129   "mrs    %0, cpsr        @ save_flags\n"     \
130     : "=r" (x)                          \
131     :                             \
132     : "memory")
133
134 /*
135  * restore saved IRQ & FIQ state
136  */
137 #define __restore_flags(x)                    \
138   __asm__ __volatile__(                   \
139   "msr    cpsr_c, %0      @ restore_flags\n"  \
```

```
140     :                               \
141     : "r" (x)                           \
142     : "memory")
143
144   #if defined(CONFIG_CPU_SA1100) || defined(CONFIG_CPU_SA110)
145   /*
146    * On the StrongARM, "swp" is terminally broken since it bypasses the
147    * cache totally.  This means that the cache becomes inconsistent, and,
148    * since we use normal loads/stores as well, this is really bad.
149    * Typically, this causes oopsen in filp_close, but could have other,
150    * more disasterous effects.  There are two work-arounds:
151    *  1. Disable interrupts and emulate the atomic swap
152    *  2. Clean the cache, perform atomic swap, flush the cache
153    *
154    * We choose (1) since its the "easiest" to achieve here and is not
155    * dependent on the processor type.
156    */
157   #define swp_is_buggy
158   #endif
159
160   extern __inline__ unsigned long __xchg(unsigned long x, volatile void *ptr, int
size)
161   {
162     extern void __bad_xchg(volatile void *, int);
163     unsigned long ret;
164   #ifdef swp_is_buggy
165     unsigned long flags;
166   #endif
167
168     switch (size) {
169   #ifdef swp_is_buggy
170         case 1:
171             __save_flags_cli(flags);
172             ret = *(volatile unsigned char *)ptr;
173             *(volatile unsigned char *)ptr = x;
174             __restore_flags(flags);
175             break;
176
177         case 4:
178             __save_flags_cli(flags);
179             ret = *(volatile unsigned long *)ptr;
180             *(volatile unsigned long *)ptr = x;
181             __restore_flags(flags);
182             break;
183   #else
184         case 1: __asm__ __volatile__ ("swpb %0, %1, [%2]"
185                     : "=r" (ret)
```

```
186                     : "r" (x), "r" (ptr)
187                     : "memory");
188             break;
189         case 4: __asm__ __volatile__ ("swp %0, %1, [%2]"
190                     : "=r" (ret)
191                     : "r" (x), "r" (ptr)
192                     : "memory");
193             break;
194   #endif
195         default: __bad_xchg(ptr, size);
196     }
197
198     return ret;
199   }
200
201   #endif
```

uClinux-dist/linux-2.4.x/drivers/char/Config.in

```
#
# Character device configuration
#
mainmenu_option next_comment
comment 'Character devices'

#############################################################################
#
# uClinux options
 #

 if [ "$CONFIG_EXCALIBUR" = "y" ]; then
   bool 'Nios serial support' CONFIG_NIOS_SERIAL
   bool 'Nios SPI device support' CONFIG_NIOS_SPI
 fi

 if [ "$CONFIG_M68328" = "y" ]; then
   bool '68328 serial support' CONFIG_68328_SERIAL
   if [ "$CONFIG_68328_SERIAL" = "y" ]; then
     bool 'Support RTS/CTS on 68328 serial support' CONFIG_68328_SERIAL_RTS_CTS
   fi
   if [ "$CONFIG_PILOT" = "y" ]; then
     bool '68328 digitizer support' CONFIG_68328_DIGI
   fi
 fi

 if [ "$CONFIG_M68EZ328" = "y" ]; then
   bool '68328 serial support' CONFIG_68328_SERIAL
   if [ "$CONFIG_68328_SERIAL" = "y" ]; then
     bool 'Support RTS/CTS on 68328 serial support' CONFIG_68328_SERIAL_RTS_CTS
   fi
   if [ "$CONFIG_M68EZ328ADS" = "y" ]; then
     bool '68681 serial support' CONFIG_68681_SERIAL
   fi
   bool '68328 digitizer support' CONFIG_68328_DIGI
 fi

 if [ "$CONFIG_M68VZ328" = "y" ]; then
   bool '68328 serial support' CONFIG_68328_SERIAL
   if [ "$CONFIG_68328_SERIAL" = "y" ]; then
              bool 'Enable RTS/CTS support on 1st 68328 serial port'
CONFIG_68328_SERIAL_RTS_CTS
     bool 'Enable second 68328 serial support' CONFIG_68328_SERIAL_UART2
     if [ "$CONFIG_68328_SERIAL_UART2" = "y" ]; then
                  bool 'Enable RTS/CTS support on 2nd 68328 serial port'
```

```
CONFIG_68328_SERIAL_UART2_RTS_CTS
45        fi
46      fi
47      if [ "$CONFIG_DRAGONIXVZ" = "y"]; then
48        bool 'Dragonix VZ SPI driver' CONFIG_DRAGONIX_SPI
49      fi
50    fi
51
52    if [ "$CONFIG_M68332" = "y" ]; then
53      bool '68332 serial support' CONFIG_68332_SERIAL
54    fi
55
56    if [ "$CONFIG_M68EN302" = "y" ]; then
57      bool '68302 serial support' CONFIG_68302_SERIAL
58    fi
59
60    if [ "$CONFIG_360QUICC" = "y" ];then
61      bool 'Support UART on MC68360 SMC' CONFIG_M68360_SMC_UART
62      bool 'Support UART on MC68360 SCC' CONFIG_M68360_SCC_UART
63      if [ "$CONFIG_M68360_SMC_UART" = "y" -o "$CONFIG_M68360_SCC_UART" = "y" ]; then
64          define_bool CONFIG_M68360_UART y
65      fi
66      if [ "$CONFIG_M68360_SMC_UART" = "y" ]; then
67        bool '68360 Serial console' CONFIG_SERIAL_CONSOLE
68        # This is entirely the wrong way to do this :(
69        if [ "$CONFIG_SERIAL_CONSOLE" = "y" ]; then
70          choice 'Initial serial console speed' \
71            "9600   CONFIG_CONSOLE_9600 \
72            19200  CONFIG_CONSOLE_19200 \
73            115200 CONFIG_CONSOLE_115200" 19200
74        fi
75      fi
76    fi
77
78    if [ "$CONFIG_COLDFIRE" = "y" ]; then
79      bool 'ColdFire serial support' CONFIG_COLDFIRE_SERIAL
80      bool 'ColdFire MBUS Support' CONFIG_MCF_MBUS
81      tristate 'ColdFire QSPI Support' CONFIG_MCF_QSPI
82      bool 'Support for TEXT based LCD driver' CONFIG_LCDTXT
83      bool 'Support for ColdFire DMA driven LCD driver' CONFIG_LCDDMA
84      bool 'Support for ColdFire DMA driven DAC0800 driver' CONFIG_DAC0800
85      bool 'Support for ColdFire 5249 audio' CONFIG_M5249AUDIO
86      bool 'Support for ColdFire DMA driven I2S DAC driver' CONFIG_DACI2S
87      bool 'Support for ColdFire T6963 driver' CONFIG_T6963
88      if [ "$CONFIG_T6963" = "y" ]; then
89        bool '   Use PIO mode' CONFIG_T6963_PIO
90        if [ "$CONFIG_T6963_PIO" != "y" ]; then
```

```
91          define_bool CONFIG_T6963_DMA y
92        fi
93      fi
94      bool 'Expansion interface driver' CONFIG_EXP
95      bool 'Reset switch support' CONFIG_RESETSWITCH
96      bool 'ColdFire Watchdog Timer Support' CONFIG_MCFWATCHDOG
97    fi
98
99    if [ "$CONFIG_ARCH_INTEGRATOR" = "y" ]; then
100       bool 'ARM AMBA serial port support' CONFIG_SERIAL_AMBA
101       if [ "$CONFIG_SERIAL_AMBA" = "y" ]; then
102          define_bool CONFIG_SERIAL_INTEGRATOR y
103                  bool '    Support for console on Integrator serial port'
CONFIG_SERIAL_AMBA_CONSOLE
104       fi
105    fi
106
107    if [ "$CONFIG_BOARD_SNDS100" = "y" ]; then
108       bool 'Samsung serial port support' CONFIG_SERIAL_SAMSUNG
109       if [ "$CONFIG_SERIAL_SAMSUNG" = "y" ]; then
110                   bool '    Support for console on Samsung serial port'
CONFIG_SERIAL_SAMSUNG_CONSOLE
111       fi
112    fi
113
114    if [ "$CONFIG_ARCH_S3C44B0" = "y" ]; then
115       bool 'S3C44B0 serial port support' CONFIG_SERIAL_S3C44B0
116       if [ "$CONFIG_SERIAL_S3C44B0" = 'y' ]; then
117                    bool '  Support for console on S3C44B0 serial port'
CONFIG_SERIAL_S3C44B0_CONSOLE
118       fi
119    fi
120
121    if [ "$CONFIG_CPU_S3C3410" = "y" ]; then
122       bool 'Samsung S3C3410X serial ports support' CONFIG_SERIAL_S3C3410
123       if [ "$CONFIG_SERIAL_S3C3410" = "y" ]; then
124          choice 'Serial console device' \
125            "NULL   CONFIG_CONSOLE_NULL \
126             UART   CONFIG_CONSOLE_UART" UART
127          if [ "$CONFIG_CONSOLE_NULL" = "n" ]; then
128             define_bool CONFIG_SERIAL_S3C3410_CONSOLE y
129             if [ "$CONFIG_UCBOOTSTRAP" = "n" ]; then
130                int '   Initial serial console speed' CONFIG_INIT_CONSOLE_SPEED
131             fi
132          fi
133       fi
134    fi
```

```
135
136   if [ "$CONFIG_CPU_S3C4530" = "y" ]; then
137      bool 'Samsung S3C4530a serial ports support' CONFIG_SERIAL_S3C4530
138      if [ "$CONFIG_SERIAL_S3C4530" = "y" ]; then
139         choice 'Serial console device' \
140           "NULL   CONFIG_CONSOLE_NULL \
141            UARTA  CONFIG_CONSOLE_UARTA \
142            UARTB  CONFIG_CONSOLE_UARTB" UARTA
143         if [ "$CONFIG_CONSOLE_NULL" = "n" ]; then
144            define_bool CONFIG_SERIAL_S3C4530_CONSOLE y
145            if [ "$CONFIG_UCBOOTSTRAP" = "n" ]; then
146               int '   Initial serial console speed' CONFIG_INIT_CONSOLE_SPEED
147            fi
148         fi
149      fi
150   fi
151
152   if [ "$CONFIG_ARCH_DSC21" = "y" ]; then
153      bool 'DSC21 serial port support' CONFIG_SERIAL_DSC21
154      if [ "$CONFIG_SERIAL_DSC21" = "y" ]; then
155         bool 'Console on DSC21 serial port' CONFIG_SERIAL_DSC21_CONSOLE
156   #      if [ "$CONFIG_SERIAL_DSC21_CONSOLE" = "y" ]; then
157   #         int 'Console UART (0 or 1)' CONFIG_SERIAL_DSC21_CONSOLE_UART
158   #      fi
159      fi
160   fi
161
162   if [ "$CONFIG_ARCH_NETARM" = "y" ]; then
163      bool 'NET+ARM serial port support' CONFIG_SERIAL_NETARM
164      if [ "$CONFIG_SERIAL_NETARM" = "y" ]; then
165         bool 'Console on NETARM serial port' CONFIG_SERIAL_NETARM_CONSOLE
166      fi
167   fi
168
169   if [ "$CONFIG_CPU_H8300H" = "y" ]; then
170      bool 'Serial (SCI) support' CONFIG_SH_SCI
171      if [ "$CONFIG_SH_SCI" = "y" ]; then
172         bool '  Support for console on serial port' CONFIG_SERIAL_CONSOLE
173      fi
174   fi
175
176   bool 'LED Manager support' CONFIG_LEDMAN
177   bool 'DS1302 Real Time Clock support' CONFIG_DS1302
178
179
#############################################################################
180
```

```
181   bool 'Virtual terminal' CONFIG_VT
182   if [ "$CONFIG_VT" = "y" ]; then
183      bool '  Support for console on virtual terminal' CONFIG_VT_CONSOLE
184      if [ "$CONFIG_GSC_LASI" = "y" ]; then
185         bool '    Support for Lasi/Dino PS2 port' CONFIG_GSC_PS2
186      fi
187   fi
188   tristate 'Standard/generic (8250/16550 and compatible UARTs) serial support'
CONFIG_SERIAL
189   if [ "$CONFIG_SERIAL" = "y" ]; then
190      bool '  Support for console on serial port' CONFIG_SERIAL_CONSOLE
191      if [ "$CONFIG_GSC_LASI" = "y" ]; then
192         bool '   serial port on GSC support' CONFIG_SERIAL_GSC
193      fi
194      if [ "$CONFIG_IA64" = "y" ]; then
195           bool 'Support for serial console port described by EFI HCDP table'
CONFIG_SERIAL_HCDP
196      fi
197      if [ "$CONFIG_ARCH_ACORN" = "y" ]; then
198         tristate '   Atomwide serial port support' CONFIG_ATOMWIDE_SERIAL
199         tristate '   Dual serial port support' CONFIG_DUALSP_SERIAL
200      fi
201   fi
202     dep_mbool 'Extended dumb serial driver options' CONFIG_SERIAL_EXTENDED
$CONFIG_SERIAL
203   if [ "$CONFIG_SERIAL_EXTENDED" = "y" ]; then
204      bool '  Support more than 4 serial ports' CONFIG_SERIAL_MANY_PORTS
205      bool '  Support for sharing serial interrupts' CONFIG_SERIAL_SHARE_IRQ
206      bool '  Autodetect IRQ on standard ports (unsafe)' CONFIG_SERIAL_DETECT_IRQ
207      bool '  Support special multiport boards' CONFIG_SERIAL_MULTIPORT
208      bool '  Support the Bell Technologies HUB6 card' CONFIG_HUB6
209   fi
210   bool 'Non-standard serial port support' CONFIG_SERIAL_NONSTANDARD
211   if [ "$CONFIG_SERIAL_NONSTANDARD" = "y" ]; then
212      tristate '  Computone IntelliPort Plus serial support' CONFIG_COMPUTONE
213      tristate '  Comtrol Rocketport support' CONFIG_ROCKETPORT
214      tristate '  Cyclades async mux support' CONFIG_CYCLADES
215      if [ "$CONFIG_EXPERIMENTAL" = "y" -a "$CONFIG_CYCLADES" != "n" ]; then
216              bool '      Cyclades-Z interrupt mode operation (EXPERIMENTAL)'
CONFIG_CYZ_INTR
217      fi
218      if [ "$CONFIG_X86_64" != "y" ]; then
219         tristate '  Digiboard Intelligent Async Support' CONFIG_DIGIEPCA
220         if [ "$CONFIG_DIGIEPCA" = "n" ]; then
221            tristate '  Digiboard PC/Xx Support' CONFIG_DIGI
222         fi
223      fi
```

```
224      dep_tristate '  Hayes ESP serial port support' CONFIG_ESPSERIAL $CONFIG_ISA
225      tristate '  Moxa Intellio support' CONFIG_MOXA_INTELLIO
226      tristate '  Moxa SmartIO support' CONFIG_MOXA_SMARTIO
227      if [ "$CONFIG_EXPERIMENTAL" = "y" ]; then
228           dep_tristate '  Multi-Tech multiport card support (EXPERIMENTAL)'
CONFIG_ISI m
229      fi
230      tristate '  Microgate SyncLink card support' CONFIG_SYNCLINK
231      tristate '  SyncLink Multiport support' CONFIG_SYNCLINKMP
232      tristate '  HDLC line discipline support' CONFIG_N_HDLC
233      tristate '  SDL RISCom/8 card support' CONFIG_RISCOM8
234      if [ "$CONFIG_X86_64" != "y" ]; then
235         tristate '  Specialix IO8+ card support' CONFIG_SPECIALIX
236         if [ "$CONFIG_SPECIALIX" != "n" ]; then
237            bool '  Specialix DTR/RTS pin is RTS' CONFIG_SPECIALIX_RTSCTS
238         fi
239         tristate '  Specialix SX (and SI) card support' CONFIG_SX
240         tristate '  Specialix RIO system support' CONFIG_RIO
241         if [ "$CONFIG_RIO" != "n" ]; then
242           bool '    Support really old RIO/PCI cards' CONFIG_RIO_OLDPCI
243         fi
244      fi
245      bool '  Stallion multiport serial support' CONFIG_STALDRV
246      if [ "$CONFIG_STALDRV" = "y" ]; then
247        tristate '    Stallion EasyIO or EC8/32 support' CONFIG_STALLION
248        tristate '    Stallion EC8/64, ONboard, Brumby support' CONFIG_ISTALLION
249      fi
250      if [ "$CONFIG_MIPS" = "y" ]; then
251        bool '  TX3912/PR31700 serial port support' CONFIG_SERIAL_TX3912
252               dep_bool '          Console on TX3912/PR31700 serial port'
CONFIG_SERIAL_TX3912_CONSOLE $CONFIG_SERIAL_TX3912
253        bool '  Enable Au1000 UART Support' CONFIG_AU1000_UART
254        if [ "$CONFIG_AU1000_UART" = "y" ]; then
255                    bool '              Enable Au1000 serial console'
CONFIG_AU1000_SERIAL_CONSOLE
256        fi
257        bool 'TXx927 SIO support' CONFIG_TXX927_SERIAL
258        if [ "$CONFIG_TXX927_SERIAL" = "y" ]; then
259           bool 'TXx927 SIO Console support' CONFIG_TXX927_SERIAL_CONSOLE
260        fi
261        if [ "$CONFIG_SIBYTE_SB1250" = "y" ]; then
262           bool '  Support for sb1250 onchip DUART' CONFIG_SIBYTE_SB1250_DUART
263           if [ "$CONFIG_SIBYTE_SB1250_DUART" = "y" ]; then
264              bool '  Console on SB1250 DUART' CONFIG_SIBYTE_SB1250_DUART_CONSOLE
265        if [ "$CONFIG_SIBYTE_SB1250_DUART_CONSOLE" = "y" ]; then
266           define_bool CONFIG_SERIAL_CONSOLE y
267        fi
```

```
268        int          '          Output     buffers     size     (in     bytes)'
CONFIG_SB1250_DUART_OUTPUT_BUF_SIZE 1024
269        bool    '       Leave    port    1    alone    (for    kgdb    or    audio)'
CONFIG_SIBYTE_SB1250_DUART_NO_PORT_1
270          fi
271        fi
272     fi
273   fi
274   if [ "$CONFIG_EXPERIMENTAL" = "y" -a "$CONFIG_ZORRO" = "y" ]; then
275      tristate 'Commodore A2232 serial support (EXPERIMENTAL)' CONFIG_A2232
276   fi
277   if [ "$CONFIG_FOOTBRIDGE" = "y" ]; then
278      bool 'DC21285 serial port support' CONFIG_SERIAL_21285
279      if [ "$CONFIG_SERIAL_21285" = "y" ]; then
280         if [ "$CONFIG_OBSOLETE" = "y" ]; then
281            bool '  Use /dev/ttyS0 device (OBSOLETE)' CONFIG_SERIAL_21285_OLD
282         fi
283         bool '  Console on DC21285 serial port' CONFIG_SERIAL_21285_CONSOLE
284      fi
285      if [ "$CONFIG_MIPS" = "y" ]; then
286        bool '  TMPTX3912/PR31700 serial port support' CONFIG_SERIAL_TX3912
287              dep_bool '          Console  on  TMPTX3912/PR31700  serial  port'
CONFIG_SERIAL_TX3912_CONSOLE $CONFIG_SERIAL_TX3912
288        bool '  Enable Au1000 UART Support' CONFIG_AU1000_UART
289        if [ "$CONFIG_AU1000_UART" = "y" ]; then
290                        bool '                Enable  Au1000  serial  console'
CONFIG_AU1000_SERIAL_CONSOLE
291        fi
292      fi
293      if [ "$CONFIG_PARISC" = "y" ]; then
294        bool '  PDC software console support' CONFIG_PDC_CONSOLE
295      fi
296   fi
297   if [ "$CONFIG_MIPS_ITE8172" = "y" ]; then
298      bool 'Enable Qtronix 990P Keyboard Support' CONFIG_QTRONIX_KEYBOARD
299      if [ "$CONFIG_QTRONIX_KEYBOARD" = "y" ]; then
300        define_bool CONFIG_IT8172_CIR y
301      else
302        bool '    Enable PS2 Keyboard Support' CONFIG_PC_KEYB
303      fi
304      bool 'Enable Smart Card Reader 0 Support ' CONFIG_IT8172_SCR0
305      bool 'Enable Smart Card Reader 1 Support ' CONFIG_IT8172_SCR1
306   fi
307   if [ "$CONFIG_MIPS_IVR" = "y" ]; then
308      bool 'Enable Qtronix 990P Keyboard Support' CONFIG_QTRONIX_KEYBOARD
309      if [ "$CONFIG_QTRONIX_KEYBOARD" = "y" ]; then
310        define_bool CONFIG_IT8172_CIR y
```

```
311       fi
312       bool 'Enable Smart Card Reader 0 Support ' CONFIG_IT8172_SCR0
313    fi
314    bool 'Unix98 PTY support' CONFIG_UNIX98_PTYS
315    if [ "$CONFIG_UNIX98_PTYS" = "y" ]; then
316       int 'Maximum number of Unix98 PTYs in use (0-2048)' CONFIG_UNIX98_PTY_COUNT
256
317    fi
318    if [ "$CONFIG_PARPORT" != "n" ]; then
319       dep_tristate 'Parallel printer support' CONFIG_PRINTER $CONFIG_PARPORT
320       if [ "$CONFIG_PRINTER" != "n" ]; then
321          bool '  Support for console on line printer' CONFIG_LP_CONSOLE
322       fi
323          dep_tristate 'Support for user-space parallel port device drivers'
CONFIG_PPDEV $CONFIG_PARPORT
324    fi
325
326    if [ "$CONFIG_PPC64" ] ; then
327       bool 'pSeries Hypervisor Virtual Console support' CONFIG_HVC_CONSOLE
328    fi
329
330    source drivers/i2c/Config.in
331
332    mainmenu_option next_comment
333    comment 'Mice'
334    tristate 'Bus Mouse Support' CONFIG_BUSMOUSE
335    if [ "$CONFIG_BUSMOUSE" != "n" ]; then
336             dep_tristate '    ATIXL busmouse support' CONFIG_ATIXL_BUSMOUSE
$CONFIG_BUSMOUSE
337             dep_tristate '    Logitech busmouse support' CONFIG_LOGIBUSMOUSE
$CONFIG_BUSMOUSE
338             dep_tristate '    Microsoft busmouse support' CONFIG_MS_BUSMOUSE
$CONFIG_BUSMOUSE
339       if [ "$CONFIG_ADB" = "y" -a "$CONFIG_ADB_KEYBOARD" = "y" ]; then
340                dep_tristate '   Apple Desktop Bus mouse support (old driver)'
CONFIG_ADBMOUSE $CONFIG_BUSMOUSE
341       fi
342    fi
343
344    tristate 'Mouse Support (not serial and bus mice)' CONFIG_MOUSE
345    if [ "$CONFIG_MOUSE" != "n" ]; then
346       bool '  PS/2 mouse (aka "auxiliary device") support' CONFIG_PSMOUSE
347          tristate '   C&T 82C710 mouse port support (as on TI Travelmate)'
CONFIG_82C710_MOUSE
348       tristate '  PC110 digitizer pad support' CONFIG_PC110_PAD
349       tristate '  MK712 touch screen support' CONFIG_MK712_MOUSE
350    fi
```

```
351   endmenu
352
353   source drivers/char/joystick/Config.in
354
355   tristate 'QIC-02 tape support' CONFIG_QIC02_TAPE
356   if [ "$CONFIG_QIC02_TAPE" != "n" ]; then
357      bool '  Do you want runtime configuration for QIC-02' CONFIG_QIC02_DYNCONF
358      if [ "$CONFIG_QIC02_DYNCONF" != "y" ]; then
359         comment '  Edit configuration parameters in ./include/linux/tpqic02.h!'
360      else
361         comment '  Setting runtime QIC-02 configuration is done with qic02conf'
362         comment '  from the tpqic02-support package.  It is available at'
363         comment '  metalab.unc.edu or ftp://titus.cfw.com/pub/Linux/util/'
364      fi
365   fi
366
367   mainmenu_option next_comment
368   comment 'Watchdog Cards'
369   bool 'Watchdog Timer Support' CONFIG_WATCHDOG
370   if [ "$CONFIG_WATCHDOG" != "n" ]; then
371      bool '  Disable watchdog shutdown on close' CONFIG_WATCHDOG_NOWAYOUT
372      tristate '  Acquire SBC Watchdog Timer' CONFIG_ACQUIRE_WDT
373      tristate '  Advantech SBC Watchdog Timer' CONFIG_ADVANTECH_WDT
374      tristate '  ALi M7101 PMU Watchdog Timer' CONFIG_ALIM7101_WDT
375      tristate '  AMD "Elan" SC520 Watchdog Timer' CONFIG_SC520_WDT
376      tristate '  Berkshire Products PC Watchdog' CONFIG_PCWATCHDOG
377      if [ "$CONFIG_FOOTBRIDGE" = "y" ]; then
378         tristate '  DC21285 watchdog' CONFIG_21285_WATCHDOG
379         if [ "$CONFIG_ARCH_NETWINDER" = "y" ]; then
380            tristate '  NetWinder WB83C977 watchdog' CONFIG_977_WATCHDOG
381         fi
382      fi
383      tristate '  Eurotech CPU-1220/1410 Watchdog Timer' CONFIG_EUROTECH_WDT
384      tristate '  IB700 SBC Watchdog Timer' CONFIG_IB700_WDT
385      tristate '  ICP ELectronics Wafer 5823 Watchdog' CONFIG_WAFER_WDT
386      if [ "$CONFIG_SGI_IP22" = "y" ]; then
387                dep_tristate '    Indy/I2  Hardware  Watchdog'  CONFIG_INDYDOG
$CONFIG_SGI_IP22
388      fi
389      tristate '  Intel i810 TCO timer / Watchdog' CONFIG_I810_TCO
390      tristate '  Mixcom Watchdog' CONFIG_MIXCOMWD
391      tristate '  SBC-60XX Watchdog Timer' CONFIG_60XX_WDT
392       dep_tristate '  SC1200 Watchdog Timer (EXPERIMENTAL)' CONFIG_SC1200_WDT
$CONFIG_EXPERIMENTAL
393      tristate '  Software Watchdog' CONFIG_SOFT_WATCHDOG
394      tristate '  W83877F (EMACS) Watchdog Timer' CONFIG_W83877F_WDT
395      tristate '  WDT Watchdog timer' CONFIG_WDT
```

```
396      tristate '  WDT PCI Watchdog timer' CONFIG_WDTPCI
397      if [ "$CONFIG_WDT" != "n" ]; then
398         bool '    WDT501 features' CONFIG_WDT_501
399         if [ "$CONFIG_WDT_501" = "y" ]; then
400            bool '      Fan Tachometer' CONFIG_WDT_501_FAN
401         fi
402      fi
403      tristate '  ZF MachZ Watchdog' CONFIG_MACHZ_WDT
404          dep_tristate '   AMD 766/768 TCO Timer/Watchdog' CONFIG_AMD7XX_TCO
$CONFIG_EXPERIMENTAL
405   fi
406   endmenu
407
408   if [ "$CONFIG_ARCH_NETWINDER" = "y" ]; then
409      tristate 'NetWinder thermometer support' CONFIG_DS1620
410      tristate 'NetWinder Button' CONFIG_NWBUTTON
411      if [ "$CONFIG_NWBUTTON" != "n" ]; then
412         bool '  Reboot Using Button' CONFIG_NWBUTTON_REBOOT
413      fi
414      tristate 'NetWinder flash support' CONFIG_NWFLASH
415   fi
416
417   if [ "$CONFIG_X86" = "y" -o "$CONFIG_X86_64" = "y" ]; then
418        dep_tristate 'AMD 768 Random Number Generator support' CONFIG_AMD_RNG
$CONFIG_PCI
419   fi
420   if [ "$CONFIG_X86" = "y" -o "$CONFIG_IA64" = "y" ]; then
421      dep_tristate 'Intel i8x0 Random Number Generator support' CONFIG_INTEL_RNG
$CONFIG_PCI
422   fi
423   dep_tristate 'AMD 76x native power management (Experimental)' CONFIG_AMD_PM768
$CONFIG_PCI
424   tristate '/dev/nvram support' CONFIG_NVRAM
425   tristate 'Enhanced Real Time Clock Support' CONFIG_RTC
426   if [ "$CONFIG_IA64" = "y" ]; then
427      bool 'EFI Real Time Clock Services' CONFIG_EFI_RTC
428   fi
429   if [ "$CONFIG_OBSOLETE" = "y" -a "$CONFIG_ALPHA_BOOK1" = "y" ]; then
430      bool 'Tadpole ANA H8 Support (OBSOLETE)'  CONFIG_H8
431   fi
432
433   tristate 'Double Talk PC internal speech card support' CONFIG_DTLK
434   tristate 'Siemens R3964 line discipline' CONFIG_R3964
435   tristate 'Applicom intelligent fieldbus card support' CONFIG_APPLICOM
436   if [ "$CONFIG_EXPERIMENTAL" = "y" -a "$CONFIG_X86" = "y" -a "$CONFIG_X86_64" !=
"y" ]; then
437           dep_tristate 'Sony Vaio Programmable I/O Control Device support
```

```
(EXPERIMENTAL)' CONFIG_SONYPI $CONFIG_PCI
438   fi
439
440   mainmenu_option next_comment
441   comment 'Ftape, the floppy tape device driver'
442   tristate 'Ftape (QIC-80/Travan) support' CONFIG_FTAPE
443   if [ "$CONFIG_FTAPE" != "n" ]; then
444      source drivers/char/ftape/Config.in
445   fi
446   endmenu
447
448   if [ "$CONFIG_GART_IOMMU" = "y" ]; then
449     bool '/dev/agpgart (AGP Support)' CONFIG_AGP
450     define_bool CONFIG_AGP_AMD_8151 y
451   else
452     tristate '/dev/agpgart (AGP Support)' CONFIG_AGP
453   fi
454   if [ "$CONFIG_AGP" != "n" ]; then
455      bool '  Intel 440LX/BX/GX and I815/I820/I830M/I830MP/I840/I845/I850/I860
support' CONFIG_AGP_INTEL
456      bool '  Intel I810/I815/I830M (on-board) support' CONFIG_AGP_I810
457      bool '  VIA chipset support' CONFIG_AGP_VIA
458      bool '  AMD Irongate, 761, and 762 support' CONFIG_AGP_AMD
459      if [ "$CONFIG_GART_IOMMU" != "y" ]; then
460         bool '  AMD 8151 support' CONFIG_AGP_AMD_8151
461      fi
462      bool '  Generic SiS support' CONFIG_AGP_SIS
463      bool '  ALI chipset support' CONFIG_AGP_ALI
464      bool '  Serverworks LE/HE support' CONFIG_AGP_SWORKS
465      if [ "$CONFIG_IA64" = "y" ]; then
466         bool '  HP ZX1 AGP support' CONFIG_AGP_HP_ZX1
467      fi
468   fi
469
470   bool 'Direct Rendering Manager (XFree86 DRI support)' CONFIG_DRM
471   if [ "$CONFIG_DRM" = "y" ]; then
472      bool '  Build drivers for old (XFree 4.0) DRM' CONFIG_DRM_OLD
473      if [ "$CONFIG_DRM_OLD" = "y" ]; then
474         comment 'DRM 4.0 drivers'
475         source drivers/char/drm-4.0/Config.in
476      else
477         comment 'DRM 4.1 drivers'
478         define_bool CONFIG_DRM_NEW y
479         source drivers/char/drm/Config.in
480      fi
481   fi
482
```

```
483   if [ "$CONFIG_HOTPLUG" = "y" -a "$CONFIG_PCMCIA" != "n" ]; then
484      source drivers/char/pcmcia/Config.in
485   fi
486   if [ "$CONFIG_MIPS_AU1000" = "y" ]; then
487      tristate ' Alchemy Au1000 GPIO device support' CONFIG_AU1000_GPIO
488      tristate ' Au1000/ADS7846 touchscreen support' CONFIG_TS_AU1000_ADS7846
489   fi
490   if [ "$CONFIG_MIPS_ITE8172" = "y" ]; then
491     tristate ' ITE GPIO' CONFIG_ITE_GPIO
492   fi
493
494   if [ "$CONFIG_X86" = "y" ]; then
495      tristate 'ACP Modem (Mwave) support' CONFIG_MWAVE
496   fi
497
498   endmenu
```

uClinux-dist/linux-2.4.x/drivers/char/Makefile

```
1   #
2   # Makefile for the kernel character device drivers.
3   #
4   # Note! Dependencies are done automagically by 'make dep', which also
5   # removes any old dependencies. DON'T put your own dependencies here
6   # unless it's something special (ie not a .c file).
7   #
8   # Note 2! The CFLAGS definitions are now inherited from the
9   # parent makes..
10   #
11
12   #
13   # This file contains the font map for the default (hardware) font
14   #
15   FONTMAPFILE = cp437.uni
16
17   O_TARGET := char.o
18
19   obj-y   += mem.o tty_io.o n_tty.o tty_ioctl.o raw.o pty.o misc.o random.o
20
21   # All of the (potential) objects that export symbols.
22   # This list comes from 'grep -l EXPORT_SYMBOL *.[hc]'.
23
24   export-objs     := busmouse.o console.o keyboard.o sysrq.o \
25              misc.o pty.o random.o selection.o serial.o \
26              ledman.o \
27              sonypi.o tty_io.o tty_ioctl.o generic_serial.o \
28              au1000_gpio.o hp_psaux.o nvram.o
29
30   mod-subdirs    :=  joystick ftape drm drm-4.0 pcmcia
31
32   list-multi :=
33
34   KEYMAP   =defkeymap.o
35   KEYBD    =pc_keyb.o
36   CONSOLE  =console.o
37   SERIAL   =serial.o
38
39   ifeq ($(ARCH),s390)
40     KEYMAP   =
41     KEYBD    =
42     CONSOLE  =
43     SERIAL   =
44   endif
45
```

```
46  ifeq ($(ARCH),mips)
47    ifneq ($(CONFIG_PC_KEYB),y)
48      KEYBD    =
49    endif
50  endif
51
52  ifeq ($(ARCH),s390x)
53    KEYMAP   =
54    KEYBD    =
55    CONSOLE  =
56    SERIAL   =
57  endif
58
59  ifeq ($(ARCH),m68k)
60     ifdef CONFIG_AMIGA
61        KEYBD = amikeyb.o
62     else
63        ifndef CONFIG_MAC
64      KEYBD =
65        endif
66     endif
67     SERIAL   =
68  endif
69
70
71  ifeq ($(ARCH),m68knommu)
72     ifdef CONFIG_68328_SERIAL
73         KEYBD  =
74         SERIAL = 68328serial.o
75     else
76         KEYBD =
77     endif
78  endif
79
80
81  ifdef CONFIG_Q40
82    KEYBD += q40_keyb.o
83    SERIAL = serial.o
84  endif
85
86  ifdef CONFIG_APOLLO
87    KEYBD += dn_keyb.o
88  endif
89
90  ifeq ($(ARCH),parisc)
91     ifdef CONFIG_GSC_PS2
92        KEYBD   = hp_psaux.o hp_keyb.o
```

```
93      else
94         KEYBD   =
95      endif
96      ifdef CONFIG_PDC_CONSOLE
97         CONSOLE += pdc_console.o
98      endif
99    endif
100
101    ifeq ($(ARCH),arm)
102      ifneq ($(CONFIG_PC_KEYMAP),y)
103        KEYMAP   :=
104      endif
105      ifneq ($(CONFIG_PC_KEYB),y)
106        KEYBD    :=
107      endif
108      ifeq ($(CONFIG_KMI_KEYB),y)
109        KEYBD    := amba_kmi_keyb.o
110      endif
111    endif
112
113    ifeq ($(ARCH),armnommu)
114      ifneq ($(CONFIG_PC_KEYB),y)
115        KEYBD    :=
116      endif
117    endif
118
119    ifeq ($(ARCH),sh)
120      KEYMAP   =
121      KEYBD    =
122      CONSOLE  =
123      ifeq ($(CONFIG_SH_HP600),y)
124      KEYMAP   = defkeymap.o
125      KEYBD    = scan_keyb.o hp600_keyb.o
126      CONSOLE  = console.o
127      endif
128      ifeq ($(CONFIG_SH_DMIDA),y)
129      # DMIDA does not connect the HD64465 PS/2 keyboard port
130      # but we allow for USB keyboards to be plugged in.
131      KEYMAP   = defkeymap.o
132      KEYBD    = # hd64465_keyb.o pc_keyb.o
133      CONSOLE  = console.o
134      endif
135      ifeq ($(CONFIG_SH_EC3104),y)
136      KEYMAP   = defkeymap.o
137      KEYBD    = ec3104_keyb.o
138      CONSOLE  = console.o
139      endif
```

```
140     ifeq ($(CONFIG_SH_DREAMCAST),y)
141     KEYMAP   = defkeymap.o
142     KEYBD    =
143     CONSOLE  = console.o
144     endif
145     ifeq ($(CONFIG_SH_KEYWEST),y)
146       ifdef CONFIG_DUMMY_CONSOLE
147         ifdef CONFIG_FB
148           KEYMAP  = defkeymap.o
149           CONSOLE  = console.o
150         else
151           KEYMAP  = defkeymap.o
152   #       CONSOLE  = console_sci.o
153           CONSOLE  = console.o
154         endif
155       else
156         KEYMAP  = defkeymap.o
157         CONSOLE  = console.o
158       endif
159       ifdef CONFIG_HD64465
160         KEYBD   = pc_keyb_hd64465.o
161       else
162         KEYBD   = pc_keyb.o
163       endif
164     endif
165   endif
166
167   ifeq ($(ARCH),niosnommu)
168     ifneq ($(CONFIG_PC_KEYB),y)
169       KEYBD    :=
170       KEYMAP   := defkeymap.o
171     endif
172   endif
173
174   ifeq ($(CONFIG_DECSTATION),y)
175     KEYMAP   =
176     KEYBD    =
177     SERIAL   = decserial.o
178   endif
179
180   ifeq ($(CONFIG_BAGET_MIPS),y)
181     KEYBD    =
182     SERIAL   =
183   endif
184
185   ifeq ($(CONFIG_NINO),y)
186     SERIAL   =
```

```
187   endif
188
189   ifneq ($(CONFIG_SUN_SERIAL),)
190     SERIAL   =
191   endif
192
193   ifeq ($(CONFIG_QTRONIX_KEYBOARD),y)
194     KEYBD    = qtronix.o
195     KEYMAP   = qtronixmap.o
196   endif
197
198   ifeq ($(CONFIG_DUMMY_KEYB),y)
199     KEYBD = dummy_keyb.o
200   endif
201
202   obj-$(CONFIG_VT) += vt.o vc_screen.o consolemap.o consolemap_deftbl.o $(CONSOLE)
selection.o
203   obj-$(CONFIG_SERIAL) += $(SERIAL)
204   obj-$(CONFIG_SERIAL_HCDP) += hcdp_serial.o
205   obj-$(CONFIG_SERIAL_21285) += serial_21285.o
206   obj-$(CONFIG_SERIAL_SA1100) += serial_sa1100.o
207   obj-$(CONFIG_SERIAL_AMBA) += serial_amba.o
208   obj-$(CONFIG_TS_AU1000_ADS7846) += au1000_ts.o
209   obj-$(CONFIG_NIOS_SERIAL) += NIOSserial.o
210   obj-$(CONFIG_NIOS_SPI) += spi.o
211
212   ifndef CONFIG_SUN_KEYBOARD
213     obj-$(CONFIG_VT) += keyboard.o $(KEYMAP) $(KEYBD)
214   else
215     obj-$(CONFIG_PCI) += keyboard.o $(KEYMAP)
216   endif
217
218   #
219   # uClinux drivers
220   #
221   obj-$(CONFIG_68328_SERIAL) += 68328serial.o
222   obj-$(CONFIG_SERIAL_DSC21) += serial_dsc21.o
223   obj-$(CONFIG_SERIAL_ATMEL) += serial_atmel.o
224   obj-$(CONFIG_SERIAL_NETARM) += serial_netarm.o
225   obj-$(CONFIG_SERIAL_SAMSUNG) += serial_samsung.o
226   obj-$(CONFIG_SERIAL_S3C44B0) += serial_core_44b0x.o
227   obj-$(CONFIG_SERIAL_S3C44B0) += s3c44b0.o
228   obj-$(CONFIG_SERIAL_S3C3410) += serial_s3c3410.o
229   obj-$(CONFIG_SERIAL_S3C4530) += serial_s3c4530.o
230   obj-$(CONFIG_V850E_NB85E_UART) += nb85e_uart.o generic_serial.o
231   obj-$(CONFIG_COLDFIRE_SERIAL) += mcfserial.o
232   obj-$(CONFIG_MCF_QSPI) += mcf_qspi.o
```

```
233   obj-$(CONFIG_LEDMAN) += ledman.o
234   obj-$(CONFIG_LCDTXT) += lcdtxt.o
235   obj-$(CONFIG_68328_DIGI) += mc68328digi.o
236   obj-$(CONFIG_MCFWATCHDOG) += mcfwatchdog.o
237   obj-$(CONFIG_M5249AUDIO) += m5249audio.o
238   obj-$(CONFIG_DRAGONIX_SPI) += dragonixspi.o
239   obj-$(CONFIG_DS1302) += ds1302.o
240   obj-$(CONFIG_EXP) += exp.o
241
242
243   obj-$(CONFIG_HIL) += hp_keyb.o
244   obj-$(CONFIG_MAGIC_SYSRQ) += sysrq.o
245   obj-$(CONFIG_ATARI_DSP56K) += dsp56k.o
246   obj-$(CONFIG_ROCKETPORT) += rocket.o
247   obj-$(CONFIG_MOXA_SMARTIO) += mxser.o
248   obj-$(CONFIG_MOXA_INTELLIO) += moxa.o
249   obj-$(CONFIG_DIGI) += pcxx.o
250   obj-$(CONFIG_DIGIEPCA) += epca.o
251   obj-$(CONFIG_CYCLADES) += cyclades.o
252   obj-$(CONFIG_STALLION) += stallion.o
253   obj-$(CONFIG_ISTALLION) += istallion.o
254   obj-$(CONFIG_SIBYTE_SB1250_DUART) += sb1250_duart.o
255   obj-$(CONFIG_COMPUTONE) += ip2.o ip2main.o
256   obj-$(CONFIG_RISCOM8) += riscom8.o
257   obj-$(CONFIG_ISI) += isicom.o
258   obj-$(CONFIG_ESPSERIAL) += esp.o
259   obj-$(CONFIG_SYNCLINK) += synclink.o
260   obj-$(CONFIG_SYNCLINKMP) += synclinkmp.o
261   obj-$(CONFIG_N_HDLC) += n_hdlc.o
262   obj-$(CONFIG_SPECIALIX) += specialix.o
263   obj-$(CONFIG_AMIGA_BUILTIN_SERIAL) += amiserial.o
264   obj-$(CONFIG_A2232) += ser_a2232.o generic_serial.o
265   obj-$(CONFIG_SX) += sx.o generic_serial.o
266   obj-$(CONFIG_RIO) += rio/rio.o generic_serial.o
267   obj-$(CONFIG_SH_SCI) += sh-sci.o generic_serial.o
268   obj-$(CONFIG_SERIAL167) += serial167.o
269   obj-$(CONFIG_MVME147_SCC) += generic_serial.o vme_scc.o
270   obj-$(CONFIG_MVME162_SCC) += generic_serial.o vme_scc.o
271   obj-$(CONFIG_BVME6000_SCC) += generic_serial.o vme_scc.o
272   obj-$(CONFIG_HVC_CONSOLE) += hvc_console.o
273   obj-$(CONFIG_SERIAL_TX3912) += generic_serial.o serial_tx3912.o
274   obj-$(CONFIG_TXX927_SERIAL) += serial_txx927.o
275
276   subdir-$(CONFIG_RIO) += rio
277   subdir-$(CONFIG_INPUT) += joystick
278
279   obj-$(CONFIG_ATIXL_BUSMOUSE) += atixlmouse.o
```

```
obj-$(CONFIG_LOGIBUSMOUSE) += logibusmouse.o
obj-$(CONFIG_PRINTER) += lp.o

ifeq ($(CONFIG_INPUT),y)
obj-y += joystick/js.o
endif

obj-$(CONFIG_BUSMOUSE) += busmouse.o
obj-$(CONFIG_DTLK) += dtlk.o
obj-$(CONFIG_R3964) += n_r3964.o
obj-$(CONFIG_APPLICOM) += applicom.o
obj-$(CONFIG_SONYPI) += sonypi.o
obj-$(CONFIG_MS_BUSMOUSE) += msbusmouse.o
obj-$(CONFIG_82C710_MOUSE) += qpmouse.o
obj-$(CONFIG_AMIGAMOUSE) += amigamouse.o
obj-$(CONFIG_ATARIMOUSE) += atarimouse.o
obj-$(CONFIG_ADBMOUSE) += adbmouse.o
obj-$(CONFIG_PC110_PAD) += pc110pad.o
obj-$(CONFIG_MK712_MOUSE) += mk712.o
obj-$(CONFIG_RTC) += rtc.o
obj-$(CONFIG_EFI_RTC) += efirtc.o
ifeq ($(CONFIG_PPC),)
  obj-$(CONFIG_NVRAM) += nvram.o
endif
obj-$(CONFIG_TOSHIBA) += toshiba.o
obj-$(CONFIG_I8K) += i8k.o
obj-$(CONFIG_DS1620) += ds1620.o
obj-$(CONFIG_INTEL_RNG) += i810_rng.o
obj-$(CONFIG_AMD_RNG) += amd768_rng.o
obj-$(CONFIG_AMD_PM768) += amd76x_pm.o

obj-$(CONFIG_ITE_GPIO) += ite_gpio.o
obj-$(CONFIG_AU1000_GPIO) += au1000_gpio.o
obj-$(CONFIG_COBALT_LCD) += lcd.o

obj-$(CONFIG_QIC02_TAPE) += tpqic02.o

subdir-$(CONFIG_FTAPE) += ftape
subdir-$(CONFIG_DRM_OLD) += drm-4.0
subdir-$(CONFIG_DRM_NEW) += drm
subdir-$(CONFIG_PCMCIA) += pcmcia
subdir-$(CONFIG_AGP) += agp

ifeq ($(CONFIG_FTAPE),y)
obj-y       += ftape/ftape.o
endif

```

```
obj-$(CONFIG_H8) += h8.o
obj-$(CONFIG_PPDEV) += ppdev.o
obj-$(CONFIG_DZ) += dz.o
obj-$(CONFIG_NWBUTTON) += nwbutton.o
obj-$(CONFIG_NWFLASH) += nwflash.o



# Only one watchdog can succeed. We probe the hardware watchdog
# drivers first, then the softdog driver.  This means if your hardware
# watchdog dies or is 'borrowed' for some reason the software watchdog
# still gives you some cover.

obj-$(CONFIG_PCWATCHDOG) += pcwd.o
obj-$(CONFIG_ACQUIRE_WDT) += acquirewdt.o
obj-$(CONFIG_ADVANTECH_WDT) += advantechwdt.o
obj-$(CONFIG_IB700_WDT) += ib700wdt.o
obj-$(CONFIG_MIXCOMWD) += mixcomwd.o
obj-$(CONFIG_60XX_WDT) += sbc60xxwdt.o
obj-$(CONFIG_W83877F_WDT) += w83877f_wdt.o
obj-$(CONFIG_SC520_WDT) += sc520_wdt.o
obj-$(CONFIG_WDT) += wdt.o
obj-$(CONFIG_WDTPCI) += wdt_pci.o
obj-$(CONFIG_21285_WATCHDOG) += wdt285.o
obj-$(CONFIG_977_WATCHDOG) += wdt977.o
obj-$(CONFIG_I810_TCO) += i810-tco.o
obj-$(CONFIG_MACHZ_WDT) += machzwd.o
obj-$(CONFIG_SH_WDT) += shwdt.o
obj-$(CONFIG_EUROTECH_WDT) += eurotechwdt.o
obj-$(CONFIG_ALIM7101_WDT) += alim7101_wdt.o
#obj-$(CONFIG_ALIM1535_WDT) += alim1535d_wdt.o
obj-$(CONFIG_INDYDOG) += indydog.o
obj-$(CONFIG_SC1200_WDT) += sc1200wdt.o
obj-$(CONFIG_WAFER_WDT) += wafer5823wdt.o
obj-$(CONFIG_SOFT_WATCHDOG) += softdog.o
obj-$(CONFIG_AMD7XX_TCO) += amd7xx_tco.o

subdir-$(CONFIG_MWAVE) += mwave
ifeq ($(CONFIG_MWAVE),y)
  obj-y += mwave/mwave.o
endif

include $(TOPDIR)/Rules.make

fastdep:

conmakehash: conmakehash.c
```

```
374     $(HOSTCC) $(HOSTCFLAGS) -o conmakehash conmakehash.c
375
376   consolemap_deftbl.c: $(FONTMAPFILE) conmakehash
377     ./conmakehash $(FONTMAPFILE) > consolemap_deftbl.c
378
379   consolemap_deftbl.o: consolemap_deftbl.c $(TOPDIR)/include/linux/types.h
380
381   .DELETE_ON_ERROR:
382
383   defkeymap.c: defkeymap.map
384     set -e ; loadkeys --mktable $< | sed -e 's/^static *//' > $@
385
386   qtronixmap.c: qtronixmap.map
387     set -e ; loadkeys --mktable $< | sed -e 's/^static *//' > $@
```

uClinux-dist/linux-2.4.x/drivers/char/s3c44b0x.h

```
#define REGBASE     0x01c00000
#define REGL(addr)  (*(volatile unsigned int *)(REGBASE+addr))
#define REGW(addr)  (*(volatile unsigned short *)(REGBASE+addr))
#define REGB(addr)  (*(volatile unsigned char *)(REGBASE+addr))

#define rINTMSK     REGL(0x20000c)

/**********************/
/* UART Registers    */
 /**********************/

 #define rULCON0        REGL(0x100000)
 #define rULCON1        REGL(0x104000)
 #define rUCON0     REGL(0x100004)
 #define rUCON1     REGL(0x104004)
 #define rUFCON0        REGL(0x100008)
 #define rUFCON1        REGL(0x104008)
 #define rUMCON0        REGL(0x10000c)
 #define rUMCON1        REGL(0x10400c)
 #define rUTRSTAT0  REGL(0x100010)
 #define rUTRSTAT1  REGL(0x104010)
 #define rUERSTAT0  REGL(0x100014)
 #define rUERSTAT1  REGL(0x104014)
 #define rUFSTAT0       REGL(0x100018)
 #define rUFSTAT1       REGL(0x104018)
 #define rUMSTAT0       REGL(0x10001c)
 #define rUMSTAT1       REGL(0x10401c)
 #define rUTXH0     REGB(0x100020)
 #define rUTXH1     REGB(0x104020)
 #define rURXH0     REGB(0x100024)
 #define rURXH1     REGB(0x104024)
 #define rUBRDIV0       REGL(0x100028)
 #define rUBRDIV1       REGL(0x104028)
```

uClinux-dist/linux-2.4.x/drivers/char/s3c44b0x.c

```
1   /*
2    *  linux/drivers/char/serial_s3c44b0.c
3    *
4    *  Driver for S3C44B0 serial ports
5    *
6    *  Based on drivers/char/serial.c, by Linus Torvalds, Theodore Ts'o.
7    *
8    *  Copyright 1999 ARM Limited
9    *  Copyright (C) 2000 Deep Blue Solutions Ltd.
10    *
11    * This program is free software; you can redistribute it and/or modify
12    * it under the terms of the GNU General Public License as published by
13    * the Free Software Foundation; either version 2 of the License, or
14    * (at your option) any later version.
15    *
16    * This program is distributed in the hope that it will be useful,
17    * but WITHOUT ANY WARRANTY; without even the implied warranty of
18    * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19    * GNU General Public License for more details.
20    *
21    * You should have received a copy of the GNU General Public License
22    * along with this program; if not, write to the Free Software
23    * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
24    *
25    *  $Id: s3c44b0.c,v 1.9.2.1 2001/11/27 17:35:39 rmk Exp $
26    *
27    * This is a generic driver for ARM S3C44B0-type serial ports.  They
28    * have a lot of 16550-like features, but are not register compatable.
29    * Note that although they do have CTS, DCD and DSR inputs, they do
30    * not have an RI input, nor do they have DTR or RTS outputs.  If
31    * required, these have to be supplied via some other means (eg, GPIO)
32    * and hooked into this driver.
33    */
34   #include <linux/config.h>
35   #include <linux/module.h>
36   #include <linux/errno.h>
37   #include <linux/signal.h>
38   #include <linux/sched.h>
39   #include <linux/interrupt.h>
40   #include <linux/tty.h>
41   #include <linux/tty_flip.h>
42   #include <linux/major.h>
43   #include <linux/string.h>
44   #include <linux/fcntl.h>
45   #include <linux/ptrace.h>
```

```
46   #include <linux/ioport.h>
47   #include <linux/mm.h>
48   #include <linux/slab.h>
49   #include <linux/init.h>
50   #include <linux/circ_buf.h>
51   #include <linux/serial.h>
52   #include <linux/console.h>
53   #include <linux/sysrq.h>
54
55   #include <asm/system.h>
56   #include <asm/io.h>
57   #include <asm/irq.h>
58   #include <asm/uaccess.h>
59   #include <asm/bitops.h>
60
61                       #if       (defined(CONFIG_SERIAL_S3C44B0_CONSOLE)       ||
defined(CONFIG_SERIAL_SAMSUNG_CONSOLE)) && defined(CONFIG_MAGIC_SYSRQ)
62   //#if defined(CONFIG_SERIAL_SAMSUNG_CONSOLE) && defined(CONFIG_MAGIC_SYSRQ)
63   #define SUPPORT_SYSRQ
64   #endif
65
66   #include "serial_core_44b0x.h"
67   #include "s3c44b0.h"
68   #include <asm/arch/irqs.h>
69   #include <asm/hardware.h>
70
71   #define UART_NR        2
72
73   #define SERIAL_S3C44B0_MAJOR   4
74   #define SERIAL_S3C44B0_MINOR   64
75   #define SERIAL_S3C44B0_NR      UART_NR
76
77   #define CALLOUT_S3C44B0_NAME   "cua"
78   #define CALLOUT_S3C44B0_MAJOR  5
79   #define CALLOUT_S3C44B0_MINOR  64
80   #define CALLOUT_S3C44B0_NR     UART_NR
81
82   static struct tty_driver normal, callout;
83   static struct tty_struct *s3c44b0_table[UART_NR];
84                  static      struct      termios      *s3c44b0_termios[UART_NR],
*s3c44b0_termios_locked[UART_NR];
85   #ifdef SUPPORT_SYSRQ
86   static struct console s3c44b0_console;
87   #endif
88
89
90   /*
```

```
91    * Access macros for the S3C44B0 UARTs
92    */
93   #define rUFSTAT        ((port)->iobase + (void *)&rUFSTAT0 - (void *)&rULCON0)
94   #define rUERSTAT   ((port)->iobase + (void *)&rUERSTAT0 - (void *)&rULCON0)
95   #define rUMSTAT        ((port)->iobase + (void *)&rUMSTAT0 - (void *)&rULCON0)
96   #define rURXH      ((port)->iobase + (void *)&rURXH0 - (void *)&rULCON0)
97   #define rUTXH      ((port)->iobase + (void *)&rUTXH0 - (void *)&rULCON0)
98   #define rUMCON     ((port)->iobase + (void *)&rUMCON0 - (void *)&rULCON0)
99   #define rUCON      ((port)->iobase + (void *)&rUCON0 - (void *)&rULCON0)
100   #define rULCON         ((port)->iobase + (void *)&rULCON0 - (void *)&rULCON0)
101   #define rUFCON         ((port)->iobase + (void *)&rUFCON0 - (void *)&rULCON0)
102   #define rUBRDIV        ((port)->iobase + (void *)&rUBRDIV0 - (void *)&rULCON0)
103
104   #define RFCNT_MASK    0x0f
105   #define TFCNT_MASK    0xf0
106   #define TFULL     0x200
107
108   #define UART_OE       1
109   #define UART_PE       2
110   #define UART_FE       3
111   #define UART_BE       4
112   #define UART_ANY_ERR  (UART_OE|UART_PE|UART_FE)
113
114   #define UART_RTS  1
115   #define UART_CTS  1
116   #define rUCON_BRK 0x10
117
118   #define rULCON_CS5    0
119   #define rULCON_CS6    1
120   #define rULCON_CS7    2
121   #define rULCON_CS8    3
122   #define rULCON_STOPB  4
123   #define rULCON_PARITY 0x20
124   #define rULCON_PEVEN  0x08
125
126
127   static void s3c44b0uart_stop_tx(struct uart_port *port, u_int from_tty)
128   {
129     outl(0x85, rUCON);
130     INT_DISABLE(port->irq);
131   }
132
133   static void s3c44b0uart_start_tx(struct uart_port *port, u_int nonempty, u_int
from_tty)
134   {
135     if(nonempty){
136         outl(0x285, rUCON);
```

```
137         INT_ENABLE(port->irq);
138     }
139   }
140
141   static void s3c44b0uart_stop_rx(struct uart_port *port)
142   {
143     INT_DISABLE(port->irq+4);
144   }
145
146   static void s3c44b0uart_enable_ms(struct uart_port *port)
147   {
148   }
149
150   static void
151   s3c44b0uart_rx_int(int irq, void *dev_id, struct pt_regs *regs)
152   {
153     struct uart_info *info = dev_id;
154     struct tty_struct *tty = info->tty;
155     unsigned int status, uer, ch, flg, ignored = 0;
156     struct uart_port *port = info->port;
157
158     status = inl(rUFSTAT);
159     while(status&RFCNT_MASK){
160         uer = inl(rUERSTAT);
161         ch = inb(rURXH);
162
163         if(tty->flip.count >= TTY_FLIPBUF_SIZE)
164             goto ignore_char;
165         port->icount.rx++;
166
167         flg = TTY_NORMAL;
168
169         if(uer&UART_ANY_ERR)
170             goto handle_error;
171         if(uart_handle_sysrq_char(info, ch, regs))
172             goto ignore_char;
173
174     error_return:
175         *tty->flip.flag_buf_ptr++ = flg;
176         *tty->flip.char_buf_ptr++ = ch;
177         tty->flip.count++;
178     ignore_char:
179         status = inl(rUFSTAT);
180     }
181   out:
182     tty_flip_buffer_push(tty);
183     return;
```

```
184
185   handle_error:
186     if(uer&UART_PE)
187         port->icount.parity++;
188     else if(uer&UART_FE)
189         port->icount.frame++;
190     if(uer&UART_OE)
191         port->icount.overrun++;
192
193     if(uer&port->ignore_status_mask){
194         if( ++ignored > 100)
195             goto out;
196         goto ignore_char;
197     }
198     uer &= port->read_status_mask;
199
200     if(uer&UART_PE)
201         flg = TTY_PARITY;
202     else if(uer&UART_FE)
203         flg = TTY_FRAME;
204
205           if (uer&UART_OE) {
206         /*
207          * CHECK: does overrun affect the current character?
208          * ASSUMPTION: it does not.
209          */
210                   *tty->flip.flag_buf_ptr++ = flg;
211                   *tty->flip.char_buf_ptr++ = ch;
212                   tty->flip.count++;
213                   if (tty->flip.count >= TTY_FLIPBUF_SIZE)
214                           goto ignore_char;
215                   ch = 0;
216                   flg = TTY_OVERRUN;
217           }
218   #ifdef SUPPORT_SYSRQ
219           info->sysrq = 0;
220   #endif
221           goto error_return;
222
223   }
224
225   static void s3c44b0uart_tx_int(int irq, void *dev_id, struct pt_regs *regs)
226   {
227     struct uart_info *info = dev_id;
228     struct uart_port *port = info->port;
229
230     if (port->x_char) {
```

```
231         outb(port->x_char, rUTXH);
232         port->icount.tx++;
233         port->x_char = 0;
234         return;
235     }
236     if (info->xmit.head == info->xmit.tail
237         || info->tty->stopped
238         || info->tty->hw_stopped) {
239         s3c44b0uart_stop_tx(port, 0);
240         return;
241     }
242
243     while(!(inl(rUFSTAT)&TFULL)){
244         outb(info->xmit.buf[info->xmit.tail], rUTXH);
245         info->xmit.tail = (info->xmit.tail + 1) & (UART_XMIT_SIZE - 1);
246         port->icount.tx++;
247         if (info->xmit.head == info->xmit.tail)
248             break;
249     };
250
251     if (CIRC_CNT(info->xmit.head,
252                  info->xmit.tail,
253                  UART_XMIT_SIZE) < WAKEUP_CHARS)
254         uart_event(info, EVT_WRITE_WAKEUP);
255
256     if (info->xmit.head == info->xmit.tail)
257         s3c44b0uart_stop_tx(info->port, 0);
258
259 }
260
261 static u_int s3c44b0uart_tx_empty(struct uart_port *port)
262 {
263     return (inl(rUFSTAT)&TFCNT_MASK) ? 0 : TIOCSER_TEMT;
264 }
265
266 static u_int s3c44b0uart_get_mctrl(struct uart_port *port)
267 {
268     return (inl(rUMSTAT)&UART_CTS) ? TIOCM_CTS : 0;
269 }
270
271 static void s3c44b0uart_set_mctrl(struct uart_port *port, u_int mctrl)
272 {
273     unsigned int status;
274
275     status = inl(rUMCON);
276
277     if(mctrl & TIOCM_RTS)
```

```
278         status |= UART_RTS;
279     else
280         status &= ~UART_RTS;
281
282     outl(status, rUMCON);
283   }
284
285   static void s3c44b0uart_break_ctl(struct uart_port *port, int break_state)
286   {
287     int status;
288
289     status = inl(rUCON);
290     if(break_state)
291         status |= rUCON_BRK;
292     else
293         status &= ~rUCON_BRK;
294     outl(status, rUCON);
295   }
296
297   static int s3c44b0uart_startup(struct uart_port *port, struct uart_info *info)
298   {
299     int retv;
300
301     retv = request_irq(port->irq, s3c44b0uart_tx_int, 0,
302         "s3c44b0_uart_tx", info);
303     if(retv)
304         return retv;
305     retv = request_irq(port->irq+4, s3c44b0uart_rx_int, 0,
306         "s3c44b0_uart_rx", info);
307     if(retv){
308         free_irq(port->irq, info);
309         return retv;
310     }
311
312     outl(0x17, rUFCON);
313     outl(0x85, rUCON);
314     INT_ENABLE(port->irq+4);
315
316     return 0;
317   }
318
319    static void s3c44b0uart_shutdown(struct uart_port *port, struct uart_info
*info)
320   {
321     free_irq(port->irq, info);
322     free_irq(port->irq+4, info);
323     outl(0x00, rUCON);
```

```
324   }
325
326   static void s3c44b0uart_change_speed(struct uart_port *port, u_int cflag, u_int
iflag, u_int quot)
327   {
328     unsigned int flags;
329     unsigned int ulcon = 0;
330
331     switch(cflag&CSIZE){
332     case CS5: ulcon = rULCON_CS5; break;
333     case CS6: ulcon = rULCON_CS6; break;
334     case CS7: ulcon = rULCON_CS7; break;
335     default:  ulcon = rULCON_CS8; break;
336     }
337     if(cflag&CSTOPB)
338         ulcon |= rULCON_STOPB;
339     if(cflag&PARENB){
340         ulcon |= rULCON_PARITY;
341         if(!(cflag&PARODD))
342             ulcon |= rULCON_PEVEN;
343     }
344
345     port->read_status_mask = UART_OE;
346     if(iflag&INPCK)
347         port->read_status_mask |= UART_PE|UART_FE;
348
349     port->ignore_status_mask = 0;
350     if(iflag&IGNPAR)
351         port->ignore_status_mask |= UART_PE|UART_FE;
352     if(iflag&IGNBRK){
353         port->ignore_status_mask |= UART_BE;
354         if(iflag&IGNPAR)
355             port->ignore_status_mask |= UART_OE;
356     }
357
358     save_flags_cli(flags);
359     outl(ulcon, rULCON);
360     outl(quot-1, rUBRDIV);
361     restore_flags(flags);
362
363   }
364
365   static const char *s3c44b0uart_type(struct uart_port *port)
366   {
367     return port->type == PORT_S3C44B0 ? "S3C44B0" : NULL;
368   }
369
```

```
370   /*
371    * Release the memory region(s) being used by 'port'
372    */
373   static void s3c44b0uart_release_port(struct uart_port *port)
374   {
375   }
376
377   /*
378    * Request the memory region(s) being used by 'port'
379    */
380   static int s3c44b0uart_request_port(struct uart_port *port)
381   {
382     return 0;
383   }
384
385   /*
386    * Configure/autoconfigure the port.
387    */
388   static void s3c44b0uart_config_port(struct uart_port *port, int flags)
389   {
390     if(flags & UART_CONFIG_TYPE)
391         port->type = PORT_S3C44B0;
392   }
393
394   /*
395    * verify the new serial_struct (for TIOCSSERIAL).
396    */
397   static int s3c44b0uart_verify_port(struct uart_port *port, struct serial_struct
*ser)
398   {
399     return 0;
400   }
401
402   static struct uart_ops s3c44b0_pops = {
403     tx_empty:   s3c44b0uart_tx_empty,
404     set_mctrl:  s3c44b0uart_set_mctrl,
405     get_mctrl:  s3c44b0uart_get_mctrl,
406     stop_tx:    s3c44b0uart_stop_tx,
407     start_tx:   s3c44b0uart_start_tx,
408     stop_rx:    s3c44b0uart_stop_rx,
409     enable_ms:  s3c44b0uart_enable_ms,
410     break_ctl:  s3c44b0uart_break_ctl,
411     startup:    s3c44b0uart_startup,
412     shutdown:   s3c44b0uart_shutdown,
413     change_speed:   s3c44b0uart_change_speed,
414     type:       s3c44b0uart_type,
415     release_port:   s3c44b0uart_release_port,
```

```
416     request_port:   s3c44b0uart_request_port,
417     config_port:    s3c44b0uart_config_port,
418     verify_port:    s3c44b0uart_verify_port,
419   };
420
421   static struct uart_port s3c44b0_ports[UART_NR] = {
422     {
423         iobase:     (void *)&rULCON0,
424         irq:        INT_UARTTX0,    //INT_UTXD0,
425         uartclk:    fMCLK_MHz,      //60000000,
426         fifosize:   16,
427         ops:        &s3c44b0_pops,
428         flags:      ASYNC_BOOT_AUTOCONF,
429     },
430     {
431         iobase:     (void *)&rULCON1,
432         irq:        INT_UARTTX1,    //INT_UTXD1,
433         uartclk:    fMCLK_MHz,      //60000000,
434         fifosize:   16,
435         ops:        &s3c44b0_pops,
436         flags:      ASYNC_BOOT_AUTOCONF,
437     }
438   };
439
440                       #if       defined(CONFIG_SERIAL_S3C44B0_CONSOLE)        ||
defined(CONFIG_SERIAL_SAMSUNG_CONSOLE)
441   //#ifdef CONFIG_SERIAL_SAMSUNG_CONSOLE
442   static void s3c44b0uart_console_write(struct console *co, const char *s, u_int
count)
443   {
444     struct uart_port *port = s3c44b0_ports + co->index;
445     unsigned int status, intmask;
446     int i;
447
448     intmask = rINTMSK;
449     INT_DISABLE(port->irq);
450     INT_DISABLE(port->irq+4);
451
452     for (i = 0; i < count; i++) {
453         do {
454             status = inl(rUFSTAT);
455         } while (status&TFULL);
456         outb(s[i], rUTXH);
457         if (s[i] == '\n') {
458             do {
459                 status = inl(rUFSTAT);
460             } while (status&TFULL);
```

```
461             outb('\r', rUTXH);
462         }
463     }
464
465     do {
466         status = inl(rUFSTAT);
467     } while (status&TFCNT_MASK);
468     rINTMSK = intmask;
469   }
470
471   static kdev_t s3c44b0uart_console_device(struct console *co)
472   {
473     return MKDEV(SERIAL_S3C44B0_MAJOR, SERIAL_S3C44B0_MINOR+co->index);
474   }
475
476   static int s3c44b0uart_console_wait_key(struct console *co)
477   {
478     struct uart_port *port = s3c44b0_ports + co->index;
479     unsigned int status, ch;
480
481     do{
482         status = inl(rUFSTAT);
483     }while(!(status&RFCNT_MASK));
484     ch = inb(rURXH);
485
486     return ch;
487   }
488
489   static void __init
490   s3c44b0uart_console_get_options(struct uart_port *port, int *baud, int *parity,
int *bits)
491   {
492     unsigned int status, quot;
493
494     status = inl(rUCON);
495     if(!status)
496         return;
497
498     status = inl(rULCON);
499     *bits = (status&3)+5;
500
501     *parity = 'n';
502     if(status&rULCON_PARITY){
503         if(status&rULCON_PEVEN)
504             *parity = 'e';
505         else
506             *parity = 'o';
```

```
507     }
508
509     quot = inl(rUBRDIV);
510     *baud = port->uartclk/(16*(quot+1));
511
512     if(*baud>100000 && *baud<130000){
513         *baud = 115200;
514         return;
515     }
516     if(*baud>200000 && *baud<250000){
517         *baud = 230400;
518         return;
519     }
520     if(*baud>51000 && *baud<66000){
521         *baud = 57600;
522         return;
523     }
524     if(*baud>34000 && *baud<42000){
525         *baud = 38400;
526         return;
527     }
528     if(*baud>17000 && *baud<22000){
529         *baud = 19200;
530         return;
531     }
532     if(*baud>8400 && *baud<11000){
533         *baud = 9600;
534         return;
535     }
536     if(*baud>4300 && *baud<5300){
537         *baud = 4800;
538         return;
539     }
540     if(*baud>2100 && *baud<2700){
541         *baud = 2400;
542         return;
543     }
544     if(*baud>1000 && *baud<1300){
545         *baud = 1200;
546         return;
547     }
548   }
549
550   static int __init s3c44b0uart_console_setup(struct console *co, char *options)
551   {
552     struct uart_port *port;
553     int baud = 57600;   //CONFIG_S3C44B0_DEFAULT_BAUDRATE;
```

```
554     int bits = 8;
555     int parity = 'n';
556     int flow = 'n';
557 
558     /*
559      * Check whether an invalid uart number has been specified, and
560      * if so, search for the first available port that does have
561      * console support.
562     */
563     port = uart_get_console(s3c44b0_ports, UART_NR, co);
564 
565 
566     if (options)
567         uart_parse_options(options, &baud, &parity, &bits, &flow);
568     else
569         s3c44b0uart_console_get_options(port, &baud, &parity, &bits);
570 
571     outl(0x97, rUFCON);
572     outl(0x05, rUCON);
573     return uart_set_options(port, co, baud, parity, bits, flow);
574   /*
575     outl(0x97, rUFCON);
576     outl(0x03, rULCON);
577     outl(0x20, rUBRDIV);
578     outl(0x05, rUCON);
579   */
580           return 0;
581   }
582 
583   static struct console s3c44b0_console = {
584     name:       "ttyS",
585     write:      s3c44b0uart_console_write,
586     device:     s3c44b0uart_console_device,
587   //    wait_key:   s3c44b0uart_console_wait_key,
588     setup:      s3c44b0uart_console_setup,
589     flags:      CON_PRINTBUFFER,
590     index:      -1,
591   };
592 
593   //void __init s3c44b0uart_console_init(void)
594   void __init samsung_console_init(void)
595   {
596     register_console(&s3c44b0_console);
597   }
598 
599   #define S3C44B0_CONSOLE   &s3c44b0_console
600   #else
```

```
601  #define S3C44B0_CONSOLE   NULL
602
603  #endif
604
605  static struct uart_driver s3c44b0_reg = {
606    owner:          THIS_MODULE,
607    normal_major:       SERIAL_S3C44B0_MAJOR,
608  #ifdef CONFIG_DEVFS_FS
609    normal_name:        "ttyS%d",
610    callout_name:       "cua%d",
611  #else
612    normal_name:        "ttyS",
613    callout_name:       "cua",
614  #endif
615    normal_driver:      &normal,
616    callout_major:      CALLOUT_S3C44B0_MAJOR,
617    callout_driver:     &callout,
618    table:          s3c44b0_table,
619    termios:        s3c44b0_termios,
620    termios_locked:     s3c44b0_termios_locked,
621    minor:          SERIAL_S3C44B0_MINOR,
622    nr:         UART_NR,
623    port:           s3c44b0_ports,
624    cons:           S3C44B0_CONSOLE,
625  };
626
627  static int __init s3c44b0uart_init(void)
628  {
629    return uart_register_driver(&s3c44b0_reg);
630  }
631
632  static void __exit s3c44b0uart_exit(void)
633  {
634    uart_unregister_driver(&s3c44b0_reg);
635  }
636
637  module_init(s3c44b0uart_init);
638  module_exit(s3c44b0uart_exit);
639
640  EXPORT_NO_SYMBOLS;
641
642  MODULE_AUTHOR("TPU");
643  MODULE_DESCRIPTION("ARM S3C44B0 serial port driver");
644  MODULE_LICENSE("GPL");
```

uClinux-dist/linux-2.4.x/drivers/char/serial_core_44b0x.h

```
1   /*
2    *  linux/drivers/char/serial_core.h
3    *
4    *  Copyright (C) 2000 Deep Blue Solutions Ltd.
5    *
6    * This program is free software; you can redistribute it and/or modify
7    * it under the terms of the GNU General Public License as published by
8    * the Free Software Foundation; either version 2 of the License, or
9    * (at your option) any later version.
10    *
11    * This program is distributed in the hope that it will be useful,
12    * but WITHOUT ANY WARRANTY; without even the implied warranty of
13    * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14    * GNU General Public License for more details.
15    *
16    * You should have received a copy of the GNU General Public License
17    * along with this program; if not, write to the Free Software
18    * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
19    *
20    *  $Id: serial_core.h,v 1.9.2.3 2001/11/26 22:32:45 rmk Exp $
21    */
22
23   /*
24    * The type definitions.  These are from Ted Ts'o's serial.h
25    */
26   #define PORT_UNKNOWN   0
27   #define PORT_8250  1
28   #define PORT_16450 2
29   #define PORT_16550 3
30   #define PORT_16550A    4
31   #define PORT_CIRRUS    5
32   #define PORT_16650 6
33   #define PORT_16650V2   7
34   #define PORT_16750 8
35   #define PORT_STARTECH  9
36   #define PORT_16C950    10
37   #define PORT_16654 11
38   #define PORT_16850 12
39   #define PORT_RSA   13
40   #define PORT_MAX_8250  13  /* max port ID */
41
42   /*
43    * ARM specific type numbers.  These are not currently guaranteed
44    * to be implemented, and will change in the future.
45    */
```

```
46   #define PORT_AMBA  32
47   #define PORT_CLPS711X  33
48   #define PORT_SA1100    34
49   #define PORT_UART00    35
50   #define PORT_21285 37
51   #define PORT_S3C44B0   38
52
53   #ifdef __KERNEL__
54
55   #include <linux/config.h>
56   #include <linux/interrupt.h>
57   #include <linux/circ_buf.h>
58   #include <linux/serial.h>
59   #include <linux/spinlock.h>
60
61   struct uart_port;
62   struct uart_info;
63
64   /*
65    * This structure describes all the operations that can be
66    * done on the physical hardware.
67    */
68   struct uart_ops {
69      u_int   (*tx_empty)(struct uart_port *);
70      void    (*set_mctrl)(struct uart_port *, u_int mctrl);
71      u_int   (*get_mctrl)(struct uart_port *);
72      void    (*stop_tx)(struct uart_port *, u_int from_tty);
73      void    (*start_tx)(struct uart_port *, u_int nonempty, u_int from_tty);
74      void    (*stop_rx)(struct uart_port *);
75      void    (*enable_ms)(struct uart_port *);
76      void    (*break_ctl)(struct uart_port *, int ctl);
77      int (*startup)(struct uart_port *, struct uart_info *);
78      void    (*shutdown)(struct uart_port *, struct uart_info *);
79      void    (*change_speed)(struct uart_port *, u_int cflag, u_int iflag, u_int
quot);
80      void    (*pm)(struct uart_port *, u_int state, u_int oldstate);
81      int (*set_wake)(struct uart_port *, u_int state);
82
83      /*
84       * Return a string describing the type of the port
85       */
86      const char *(*type)(struct uart_port *);
87
88      /*
89       * Release IO and memory resources used by the port.
90       * This includes iounmap if necessary.
91       */
```

```
92      void    (*release_port)(struct uart_port *);
93
94      /*
95       * Request IO and memory resources used by the port.
96       * This includes iomapping the port if necessary.
97       */
98      int (*request_port)(struct uart_port *);
99      void    (*config_port)(struct uart_port *, int);
100     int (*verify_port)(struct uart_port *, struct serial_struct *);
101     int (*ioctl)(struct uart_port *, u_int, u_long);
102   };
103
104   #define UART_CONFIG_TYPE  (1 << 0)
105   #define UART_CONFIG_IRQ       (1 << 1)
106
107   struct uart_icount {
108     __u32   cts;
109     __u32   dsr;
110     __u32   rng;
111     __u32   dcd;
112     __u32   rx;
113     __u32   tx;
114     __u32   frame;
115     __u32   overrun;
116     __u32   parity;
117     __u32   brk;
118     __u32   buf_overrun;
119   };
120
121   struct uart_port {
122     u_int           iobase;         /* in/out[bwl] */
123     void            *membase;       /* read/write[bwl] */
124     u_int           irq;
125     u_int           uartclk;
126     u_char          fifosize;       /* tx fifo size */
127     u_char          x_char;
128     u_char          regshift;       /* reg offset shift */
129     u_char          iotype;         /* io access style */
130     u_char          hub6;
131     u_char          unused[7];
132     u_int           read_status_mask;
133     u_int           ignore_status_mask;
134     u_int           flags;
135     u_int           type;           /* port type */
136     struct uart_ops     *ops;
137     struct uart_icount  icount;
138     u_int           line;
```

```
139     u_long          mapbase;        /* for ioremap */
140   };
141
142   /*
143    * This is the state information which is persistent across opens.
144    * The low level driver must not to touch any elements contained
145    * within.
146    */
147   struct uart_state {
148     u_int           close_delay;
149     u_int           closing_wait;
150     u_int           custom_divisor;
151     struct termios      normal_termios;
152     struct termios      callout_termios;
153
154     int         count;
155     struct uart_info    *info;
156     struct uart_port    *port;
157
158     struct semaphore    count_sem;  /* this protects 'count' */
159   #ifdef CONFIG_PM
160     struct pm_dev       *pm;
161     struct console      *cons;      /* need this to handle cons */
162   #endif
163   };
164
165   #define UART_XMIT_SIZE 1024
166   /*
167    * This is the state information which is only valid when the port
168    * is open; it may be freed by the core driver once the device has
169    * been closed.  Either the low level driver or the core can modify
170    * stuff here.
171    */
172   struct uart_info {
173     spinlock_t      lock;
174     struct uart_port    *port;
175     struct uart_ops     *ops;
176     struct uart_state   *state;
177     struct tty_struct   *tty;
178     struct circ_buf     xmit;
179     u_int           flags;
180
181     u_int           event;
182     u_int           timeout;
183     u_int           mctrl;
184     u_int           driver_priv;
185     int         blocked_open;
```

```
186     pid_t           session;
187     pid_t           pgrp;
188
189     struct tasklet_struct   tlet;
190
191     wait_queue_head_t   open_wait;
192     wait_queue_head_t   close_wait;
193     wait_queue_head_t   delta_msr_wait;
194
195     /*
196      * List if uarts on the same IRQ line.
197      */
198     struct uart_info    *next_info;
199   /*
200    * This is placed at the end since it may not be present.
201    * Do not place any new members after here.
202    */
203   #if defined(CONFIG_SERIAL_CORE_CONSOLE) && defined(CONFIG_MAGIC_SYSRQ)
204     u_long          sysrq;      /* available for driver use */
205   #endif
206   };
207
208   /* number of characters left in xmit buffer before we ask for more */
209   #define WAKEUP_CHARS      256
210
211   #define EVT_WRITE_WAKEUP  0
212
213   struct module;
214
215   struct uart_driver {
216     struct module       *owner;
217     int          normal_major;
218     const char      *normal_name;
219     struct tty_driver   *normal_driver;
220     int          callout_major;
221     const char      *callout_name;
222     struct tty_driver   *callout_driver;
223     struct tty_struct   **table;
224     struct termios      **termios;
225     struct termios      **termios_locked;
226     int          minor;
227     int          nr;
228     struct uart_state   *state;         /* driver should pass NULL */
229     struct uart_port    *port;          /* array of port information */
230     struct console      *cons;
231   };
232
```

```
233   void uart_event(struct uart_info *info, int event);
234   struct uart_port *uart_get_console(struct uart_port *ports, int nr,
235                    struct console *c);
236   void uart_parse_options(char *options, int *baud, int *parity, int *bits,
237             int *flow);
238   int uart_set_options(struct uart_port *port, struct console *co, int baud,
239              int parity, int bits, int flow);
240   int uart_register_driver(struct uart_driver *uart);
241   void uart_unregister_driver(struct uart_driver *uart);
242   void uart_unregister_port(struct uart_driver *reg, int line);
243   int uart_register_port(struct uart_driver *reg, struct uart_port *port);
244
245   /*
246    * The following are helper functions for the low level drivers.
247    */
248   #ifdef SUPPORT_SYSRQ
249   static inline int
250   __uart_handle_sysrq_char(struct uart_info *info, unsigned int ch,
251              struct pt_regs *regs)
252   {
253     if (info->sysrq) {
254         if (ch && time_before(jiffies, info->sysrq)) {
255             handle_sysrq(ch, regs, NULL, NULL);
256             info->sysrq = 0;
257             return 1;
258         }
259         info->sysrq = 0;
260     }
261     return 0;
262   }
263   #endif
264
265   /*
266    * We do the SysRQ and SAK checking like this...
267    */
268   static inline int __uart_handle_break(struct uart_info *info, struct console
*con)
269   {
270   #ifdef SUPPORT_SYSRQ
271     if (info->port->line == con->index) {
272         if (!info->sysrq) {
273             info->sysrq = jiffies + HZ*5;
274             return 1;
275         }
276         info->sysrq = 0;
277     }
278   #endif
```

```
279     if (info->flags & ASYNC_SAK)
280         do_SAK(info->tty);
281     return 0;
282   }
283
284   #ifdef SUPPORT_SYSRQ
285   #define uart_handle_break(info,con)        __uart_handle_break(info,con)
286                             #define          uart_handle_sysrq_char(info,ch,regs)
    __uart_handle_sysrq_char(info,ch,regs)
287   #else
288   #define uart_handle_break(info,con)    __uart_handle_break(info,NULL)
289   #define uart_handle_sysrq_char(info,ch,regs)   (0)
290   #endif
291
292   /**
293    *    uart_handle_dcd_change - handle a change of carrier detect state
294    *    @info: uart_info structure for the open port
295    *    @status: new carrier detect status, nonzero if active
296    */
297   static inline void
298   uart_handle_dcd_change(struct uart_info *info, unsigned int status)
299   {
300     struct uart_port *port = info->port;
301
302     port->icount.dcd++;
303
304   #ifdef CONFIG_HARD_PPS
305     if ((info->flags & ASYNC_HARDPPS_CD) && status)
306         hardpps();
307   #endif
308
309     if (info->flags & ASYNC_CHECK_CD) {
310         if (status)
311             wake_up_interruptible(&info->open_wait);
312         else if (!((info->flags & ASYNC_CALLOUT_ACTIVE) &&
313                (info->flags & ASYNC_CALLOUT_NOHUP))) {
314             if (info->tty)
315                 tty_hangup(info->tty);
316         }
317     }
318   }
319
320   /**
321    *    uart_handle_cts_change - handle a change of clear-to-send state
322    *    @info: uart_info structure for the open port
323    *    @status: new clear to send status, nonzero if active
324    */
```

```
325  static inline void
326  uart_handle_cts_change(struct uart_info *info, unsigned int status)
327  {
328    struct uart_port *port = info->port;
329    unsigned long flags;
330
331    port->icount.cts++;
332
333    if (info->flags & ASYNC_CTS_FLOW) {
334        spin_lock_irqsave(&info->lock, flags);
335        if (info->tty->hw_stopped) {
336            if (status) {
337                info->tty->hw_stopped = 0;
338                info->ops->start_tx(port, 1, 0);
339                uart_event(info, EVT_WRITE_WAKEUP);
340            }
341        } else {
342            if (!status) {
343                info->tty->hw_stopped = 1;
344                info->ops->stop_tx(port, 0);
345            }
346        }
347        spin_unlock_irqrestore(&info->lock, flags);
348    }
349  }
350
351  #endif
```

uClinux-dist/linux-2.4.x/drivers/serial_core_44b0x.c

```
1   /*
2    *  linux/drivers/char/serial_core.c
3    *
4    *  Driver core for serial ports
5    *
6    *  Based on drivers/char/serial.c, by Linus Torvalds, Theodore Ts'o.
7    *
8    *  Copyright 1999 ARM Limited
9    *  Copyright (C) 2000-2001 Deep Blue Solutions Ltd.
10    *
11    * This program is free software; you can redistribute it and/or modify
12    * it under the terms of the GNU General Public License as published by
13    * the Free Software Foundation; either version 2 of the License, or
14    * (at your option) any later version.
15    *
16    * This program is distributed in the hope that it will be useful,
17    * but WITHOUT ANY WARRANTY; without even the implied warranty of
18    * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19    * GNU General Public License for more details.
20    *
21    * You should have received a copy of the GNU General Public License
22    * along with this program; if not, write to the Free Software
23    * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
24    *
25    *  $Id: serial_core.c,v 1.20.2.5 2002/03/13 15:22:26 rmk Exp $
26    *
27    */
28   #include <linux/config.h>
29   #include <linux/module.h>
30   #include <linux/errno.h>
31   #include <linux/signal.h>
32   #include <linux/sched.h>
33   #include <linux/interrupt.h>
34   #include <linux/tty.h>
35   #include <linux/tty_flip.h>
36   #include <linux/major.h>
37   #include <linux/string.h>
38   #include <linux/fcntl.h>
39   #include <linux/ptrace.h>
40   #include <linux/ioport.h>
41   #include <linux/mm.h>
42   #include <linux/slab.h>
43   #include <linux/init.h>
44   #include <linux/circ_buf.h>
45   #include <linux/console.h>
```

```
46  #include <linux/sysrq.h>
47  #include <linux/pm.h>
48  #include "serial_core_44b0x.h"
49
50  #include <asm/system.h>
51  #include <asm/io.h>
52  #include <asm/irq.h>
53  #include <asm/uaccess.h>
54  #include <asm/bitops.h>
55
56  #undef DEBUG
57
58  #ifndef CONFIG_PM
59  #define pm_access(pm)      do { } while (0)
60  #define pm_unregister(pm)  do { } while (0)
61  #endif
62
63  /*
64   * tmp_buf is used as a temporary buffer by serial_write.  We need to
65   * lock it in case the copy_from_user blocks while swapping in a page,
66   * and some other program tries to do a serial write at the same time.
67   * Since the lock will only come under contention when the system is
68   * swapping and available memory is low, it makes sense to share one
69   * buffer across all the serial ports, since it significantly saves
70   * memory if large numbers of serial ports are open.
71   */
72  static u_char *tmp_buf;
73  static DECLARE_MUTEX(tmp_buf_sem);
74
75  /*
76   * This is used to lock changes in serial line configuration.
77   */
78  static DECLARE_MUTEX(port_sem);
79
80  #define HIGH_BITS_OFFSET   ((sizeof(long)-sizeof(int))*8)
81
82     static  void  uart_change_speed(struct  uart_info  *info,  struct  termios
*old_termios);
83  static void uart_wait_until_sent(struct tty_struct *tty, int timeout);
84
85  /*
86   * This routine is used by the interrupt handler to schedule processing in
87   * the software interrupt portion of the driver.  It is expected that
88   * interrupts will be disabled (and so the tasklet will be prevented
89   * from running (CHECK)).
90   */
91  void uart_event(struct uart_info *info, int event)
```

```
92  {
93      info->event |= 1 << event;
94      tasklet_schedule(&info->tlet);
95  }
96
97  static void uart_stop(struct tty_struct *tty)
98  {
99      struct uart_info *info = tty->driver_data;
100     unsigned long flags;
101
102     spin_lock_irqsave(&info->lock, flags);
103     info->ops->stop_tx(info->port, 1);
104     spin_unlock_irqrestore(&info->lock, flags);
105  }
106
107  static void __uart_start(struct tty_struct *tty)
108  {
109     struct uart_info *info = tty->driver_data;
110     if (info->xmit.head != info->xmit.tail && info->xmit.buf &&
111         !tty->stopped && !tty->hw_stopped)
112         info->ops->start_tx(info->port, 1, 1);
113  }
114
115  static void uart_start(struct tty_struct *tty)
116  {
117     struct uart_info *info = tty->driver_data;
118     unsigned long flags;
119
120     pm_access(info->state->pm);
121
122     spin_lock_irqsave(&info->lock, flags);
123     __uart_start(tty);
124     spin_unlock_irqrestore(&info->lock, flags);
125  }
126
127  static void uart_tasklet_action(unsigned long data)
128  {
129     struct uart_info *info = (struct uart_info *)data;
130     struct tty_struct *tty;
131
132     tty = info->tty;
133     if (!tty || !test_and_clear_bit(EVT_WRITE_WAKEUP, &info->event))
134         return;
135
136     if ((tty->flags & (1 << TTY_DO_WRITE_WAKEUP)) &&
137         tty->ldisc.write_wakeup)
138         (tty->ldisc.write_wakeup)(tty);
```

```
139     wake_up_interruptible(&tty->write_wait);
140   }
141
142   static inline void uart_update_altspeed(struct uart_info *info)
143   {
144     if ((info->flags & ASYNC_SPD_MASK) == ASYNC_SPD_HI)
145         info->tty->alt_speed = 57600;
146     if ((info->flags & ASYNC_SPD_MASK) == ASYNC_SPD_VHI)
147         info->tty->alt_speed = 115200;
148     if ((info->flags & ASYNC_SPD_MASK) == ASYNC_SPD_SHI)
149         info->tty->alt_speed = 230400;
150     if ((info->flags & ASYNC_SPD_MASK) == ASYNC_SPD_WARP)
151         info->tty->alt_speed = 460800;
152   }
153
154   static int uart_startup(struct uart_info *info)
155   {
156     unsigned long flags;
157     unsigned long page;
158     int retval = 0;
159
160     page = get_zeroed_page(GFP_KERNEL);
161     if (!page)
162         return -ENOMEM;
163
164     save_flags(flags); cli();
165
166     if (info->flags & ASYNC_INITIALIZED) {
167         free_page(page);
168         goto errout;
169     }
170
171     if (info->port->type == PORT_UNKNOWN) {
172         if (info->tty)
173             set_bit(TTY_IO_ERROR, &info->tty->flags);
174         free_page(page);
175         goto errout;
176     }
177
178     if (info->xmit.buf)
179         free_page(page);
180     else
181         info->xmit.buf = (unsigned char *) page;
182
183     info->mctrl = 0;
184
185     retval = info->ops->startup(info->port, info);
```

```
186     if (retval) {
187         if (capable(CAP_SYS_ADMIN)) {
188             if (info->tty)
189                 set_bit(TTY_IO_ERROR, &info->tty->flags);
190             retval = 0;
191         }
192         goto errout;
193     }
194 
195     if (info->tty)
196         clear_bit(TTY_IO_ERROR, &info->tty->flags);
197     info->xmit.head = info->xmit.tail = 0;
198 
199     /*
200      * Set up the tty->alt_speed kludge
201      */
202     if (info->tty)
203         uart_update_altspeed(info);
204 
205     /*
206      * and set the speed of the serial port
207      */
208     uart_change_speed(info, NULL);
209 
210     /*
211      * Setup the RTS and DTR signals once the port
212      * is open and ready to respond.
213      */
214     if (info->tty->termios->c_cflag & CBAUD)
215         info->mctrl |= TIOCM_RTS | TIOCM_DTR;
216     info->ops->set_mctrl(info->port, info->mctrl);
217 
218     info->flags |= ASYNC_INITIALIZED;
219     retval = 0;
220 
221 errout:
222     restore_flags(flags);
223     return retval;
224 }
225 
226 /*
227  * This routine will shutdown a serial port; interrupts are disabled, and
228  * DTR is dropped if the hangup on close termio flag is on.
229  */
230 static void uart_shutdown(struct uart_info *info)
231 {
232     unsigned long flags;
```

```
233
234     if (!(info->flags & ASYNC_INITIALIZED))
235         return;
236
237     save_flags(flags); cli(); /* Disable interrupts */
238
239     /*
240      * clear delta_msr_wait queue to avoid mem leaks: we may free the irq
241      * here so the queue might never be woken up
242      */
243     wake_up_interruptible(&info->delta_msr_wait);
244
245     /*
246      * Free the IRQ and disable the port
247      */
248     info->ops->shutdown(info->port, info);
249
250     if (info->xmit.buf) {
251         unsigned long pg = (unsigned long) info->xmit.buf;
252         info->xmit.buf = NULL;
253         free_page(pg);
254     }
255
256     if (!info->tty || (info->tty->termios->c_cflag & HUPCL))
257         info->mctrl &= ~(TIOCM_DTR|TIOCM_RTS);
258     info->ops->set_mctrl(info->port, info->mctrl);
259
260     /* kill off our tasklet */
261     tasklet_kill(&info->tlet);
262     if (info->tty)
263         set_bit(TTY_IO_ERROR, &info->tty->flags);
264
265     info->flags &= ~ASYNC_INITIALIZED;
266     restore_flags(flags);
267   }
268
269   static inline u_int uart_calculate_quot(struct uart_info *info, u_int baud)
270   {
271     u_int quot;
272
273     /* Special case: B0 rate */
274     if (!baud)
275         baud = 9600;
276
277     /* Old HI/VHI/custom speed handling */
278     if (baud == 38400 &&
279         ((info->flags & ASYNC_SPD_MASK) == ASYNC_SPD_CUST))
```

```
280         quot = info->state->custom_divisor;
281     else
282         quot = info->port->uartclk / (16 * baud);
283 
284     return quot;
285   }
286 
287     static  void  uart_change_speed(struct  uart_info  *info,  struct  termios
*old_termios)
288   {
289     struct uart_port *port = info->port;
290     u_int quot, baud, cflag, bits, try;
291 
292     /*
293      * If we have no tty, termios, or the port does not exist,
294      * then we can't set the parameters for this port.
295      */
296     if (!info->tty || !info->tty->termios ||
297         info->port->type == PORT_UNKNOWN)
298         return;
299 
300     cflag = info->tty->termios->c_cflag;
301 
302     /* byte size and parity */
303     switch (cflag & CSIZE) {
304     case CS5: bits = 7;  break;
305     case CS6: bits = 8;  break;
306     case CS7: bits = 9;  break;
307     default:  bits = 10; break; // CS8
308     }
309 
310     if (cflag & CSTOPB)
311         bits++;
312     if (cflag & PARENB)
313         bits++;
314 
315     for (try = 0; try < 2; try ++) {
316         /* Determine divisor based on baud rate */
317         baud = tty_get_baud_rate(info->tty);
318         quot = uart_calculate_quot(info, baud);
319         if (quot)
320             break;
321 
322         /*
323          * Oops, the quotient was zero.  Try again with
324          * the old baud rate if possible.
325          */
```

```
326         info->tty->termios->c_cflag &= ~CBAUD;
327         if (old_termios) {
328             info->tty->termios->c_cflag |=
329                   (old_termios->c_cflag & CBAUD);
330             old_termios = NULL;
331             continue;
332         }
333 
334         /*
335          * As a last resort, if the quotient is zero,
336          * default to 9600 bps
337          */
338         info->tty->termios->c_cflag |= B9600;
339     }
340 
341     info->timeout = (port->fifosize * HZ * bits * quot) /
342               (port->uartclk / 16);
343     info->timeout += HZ/50;     /* Add .02 seconds of slop */
344 
345     if (cflag & CRTSCTS)
346         info->flags |= ASYNC_CTS_FLOW;
347     else
348         info->flags &= ~ASYNC_CTS_FLOW;
349     if (cflag & CLOCAL)
350         info->flags &= ~ASYNC_CHECK_CD;
351     else
352         info->flags |= ASYNC_CHECK_CD;
353 
354     /*
355      * Set up parity check flag
356      */
357   #define RELEVENT_IFLAG(iflag) ((iflag) & (IGNBRK|BRKINT|IGNPAR|PARMRK|INPCK))
358 
359     pm_access(info->state->pm);
360 
361     info->ops->change_speed(port, cflag, info->tty->termios->c_iflag, quot);
362   }
363 
364   static void uart_put_char(struct tty_struct *tty, u_char ch)
365   {
366     struct uart_info *info = tty->driver_data;
367     unsigned long flags;
368 
369     if (!tty || !info->xmit.buf)
370         return;
371 
372     spin_lock_irqsave(&info->lock, flags);
```

```
373     if (CIRC_SPACE(info->xmit.head, info->xmit.tail, UART_XMIT_SIZE) != 0) {
374         info->xmit.buf[info->xmit.head] = ch;
375         info->xmit.head = (info->xmit.head + 1) & (UART_XMIT_SIZE - 1);
376     }
377     spin_unlock_irqrestore(&info->lock, flags);
378   }
379
380   static void uart_flush_chars(struct tty_struct *tty)
381   {
382     uart_start(tty);
383   }
384
385   static int uart_write(struct tty_struct *tty, int from_user,
386               const u_char * buf, int count)
387   {
388     struct uart_info *info = tty->driver_data;
389     unsigned long flags;
390     int c, ret = 0;
391
392     if (!tty || !info->xmit.buf || !tmp_buf)
393         return 0;
394
395     if (from_user) {
396         down(&tmp_buf_sem);
397         while (1) {
398             int c1;
399             c = CIRC_SPACE_TO_END(info->xmit.head,
400                           info->xmit.tail,
401                           UART_XMIT_SIZE);
402             if (count < c)
403                 c = count;
404             if (c <= 0)
405                 break;
406
407             c -= copy_from_user(tmp_buf, buf, c);
408             if (!c) {
409                 if (!ret)
410                     ret = -EFAULT;
411                 break;
412             }
413             spin_lock_irqsave(&info->lock, flags);
414             c1 = CIRC_SPACE_TO_END(info->xmit.head,
415                            info->xmit.tail,
416                            UART_XMIT_SIZE);
417             if (c1 < c)
418                 c = c1;
419             memcpy(info->xmit.buf + info->xmit.head, tmp_buf, c);
```

```
420             info->xmit.head = (info->xmit.head + c) &
421                       (UART_XMIT_SIZE - 1);
422             spin_unlock_irqrestore(&info->lock, flags);
423             buf += c;
424             count -= c;
425             ret += c;
426         }
427         up(&tmp_buf_sem);
428     } else {
429         spin_lock_irqsave(&info->lock, flags);
430         while (1) {
431             c = CIRC_SPACE_TO_END(info->xmit.head,
432                           info->xmit.tail,
433                           UART_XMIT_SIZE);
434             if (count < c)
435                 c = count;
436             if (c <= 0)
437                 break;
438             memcpy(info->xmit.buf + info->xmit.head, buf, c);
439             info->xmit.head = (info->xmit.head + c) &
440                       (UART_XMIT_SIZE - 1);
441             buf += c;
442             count -= c;
443             ret += c;
444         }
445         spin_unlock_irqrestore(&info->lock, flags);
446     }
447
448     uart_start(tty);
449     return ret;
450 }
451
452 static int uart_write_room(struct tty_struct *tty)
453 {
454     struct uart_info *info = tty->driver_data;
455
456     return CIRC_SPACE(info->xmit.head, info->xmit.tail, UART_XMIT_SIZE);
457 }
458
459 static int uart_chars_in_buffer(struct tty_struct *tty)
460 {
461     struct uart_info *info = tty->driver_data;
462
463     return CIRC_CNT(info->xmit.head, info->xmit.tail, UART_XMIT_SIZE);
464 }
465
466 static void uart_flush_buffer(struct tty_struct *tty)
```

```
467   {
468     struct uart_info *info = tty->driver_data;
469     unsigned long flags;
470
471   #ifdef DEBUG
472     printk("uart_flush_buffer(%d) called\n",
473            MINOR(tty->device) - tty->driver.minor_start);
474   #endif
475     spin_lock_irqsave(&info->lock, flags);
476     info->xmit.head = info->xmit.tail = 0;
477     spin_unlock_irqrestore(&info->lock, flags);
478     wake_up_interruptible(&tty->write_wait);
479     if ((tty->flags & (1 << TTY_DO_WRITE_WAKEUP)) &&
480         tty->ldisc.write_wakeup)
481         (tty->ldisc.write_wakeup)(tty);
482   }
483
484   /*
485    * This function is used to send a high-priority XON/XOFF character to
486    * the device
487    */
488   static void uart_send_xchar(struct tty_struct *tty, char ch)
489   {
490     struct uart_info *info = tty->driver_data;
491
492     info->port->x_char = ch;
493     if (ch)
494         info->ops->start_tx(info->port, 1, 0);
495   }
496
497   static void uart_throttle(struct tty_struct *tty)
498   {
499     struct uart_info *info = tty->driver_data;
500     unsigned long flags;
501
502     if (I_IXOFF(tty))
503         uart_send_xchar(tty, STOP_CHAR(tty));
504
505     if (tty->termios->c_cflag & CRTSCTS) {
506         spin_lock_irqsave(&info->lock, flags);
507         info->mctrl &= ~TIOCM_RTS;
508         info->ops->set_mctrl(info->port, info->mctrl);
509         spin_unlock_irqrestore(&info->lock, flags);
510     }
511   }
512
513   static void uart_unthrottle(struct tty_struct *tty)
```

```
514   {
515     struct uart_info *info = (struct uart_info *) tty->driver_data;
516     unsigned long flags;
517
518     if (I_IXOFF(tty)) {
519         if (info->port->x_char)
520             info->port->x_char = 0;
521         else
522             uart_send_xchar(tty, START_CHAR(tty));
523     }
524
525     if (tty->termios->c_cflag & CRTSCTS) {
526         spin_lock_irqsave(&info->lock, flags);
527         info->mctrl |= TIOCM_RTS;
528         info->ops->set_mctrl(info->port, info->mctrl);
529         spin_unlock_irqrestore(&info->lock, flags);
530     }
531   }
532
533   static int uart_get_info(struct uart_info *info, struct serial_struct *retinfo)
534   {
535     struct uart_state *state = info->state;
536     struct uart_port *port = info->port;
537     struct serial_struct tmp;
538
539     memset(&tmp, 0, sizeof(tmp));
540     tmp.type       = port->type;
541     tmp.line       = port->line;
542     tmp.port       = port->iobase;
543     if (HIGH_BITS_OFFSET)
544         tmp.port_high = port->iobase >> HIGH_BITS_OFFSET;
545     tmp.irq        = port->irq;
546     tmp.flags      = port->flags;
547     tmp.xmit_fifo_size = port->fifosize;
548     tmp.baud_base      = port->uartclk / 16;
549     tmp.close_delay    = state->close_delay;
550     tmp.closing_wait   = state->closing_wait;
551     tmp.custom_divisor = state->custom_divisor;
552     tmp.hub6       = port->hub6;
553     tmp.io_type        = port->iotype;
554     tmp.iomem_reg_shift= port->regshift;
555     tmp.iomem_base     = (void *)port->mapbase;
556
557     if (copy_to_user(retinfo, &tmp, sizeof(*retinfo)))
558         return -EFAULT;
559     return 0;
560   }
```

```
561
562   static int uart_set_info(struct uart_info *info,
563               struct serial_struct *newinfo)
564   {
565     struct serial_struct new_serial;
566     struct uart_state *state = info->state;
567     struct uart_port *port = info->port;
568     unsigned long new_port;
569     unsigned int change_irq, change_port, old_flags;
570     unsigned int old_custom_divisor;
571     int retval = 0;
572
573     if (copy_from_user(&new_serial, newinfo, sizeof(new_serial)))
574         return -EFAULT;
575
576     new_port = new_serial.port;
577     if (HIGH_BITS_OFFSET)
578         new_port += (unsigned long) new_serial.port_high << HIGH_BITS_OFFSET;
579
580     new_serial.irq = irq_cannonicalize(new_serial.irq);
581
582     /*
583      * This semaphore protects state->count.  It is also
584      * very useful to prevent opens.  Also, take the
585      * port configuration semaphore to make sure that a
586      * module insertion/removal doesn't change anything
587      * under us.
588      */
589     down(&port_sem);
590     down(&state->count_sem);
591
592     change_irq  = new_serial.irq != port->irq;
593
594     /*
595      * Since changing the 'type' of the port changes its resource
596      * allocations, we should treat type changes the same as
597      * IO port changes.
598      */
599     change_port = new_port != port->iobase ||
600               (unsigned long)new_serial.iomem_base != port->mapbase ||
601               new_serial.hub6 != port->hub6 ||
602               new_serial.io_type != port->iotype ||
603               new_serial.iomem_reg_shift != port->regshift ||
604               new_serial.type != port->type;
605
606     old_flags = port->flags;
607     old_custom_divisor = state->custom_divisor;
```

```
608
609     if (!capable(CAP_SYS_ADMIN)) {
610         retval = -EPERM;
611         if (change_irq || change_port ||
612             (new_serial.baud_base != port->uartclk / 16) ||
613             (new_serial.close_delay != state->close_delay) ||
614             (new_serial.closing_wait != state->closing_wait) ||
615             (new_serial.xmit_fifo_size != port->fifosize) ||
616             ((new_serial.flags & ~ASYNC_USR_MASK) !=
617              (port->flags & ~ASYNC_USR_MASK)))
618             goto exit;
619         port->flags = ((port->flags & ~ASYNC_USR_MASK) |
620                    (new_serial.flags & ASYNC_USR_MASK));
621         info->flags = ((info->flags & ~ASYNC_USR_MASK) |
622                    (new_serial.flags & ASYNC_USR_MASK));
623         state->custom_divisor = new_serial.custom_divisor;
624         goto check_and_exit;
625     }
626
627     /*
628      * Ask the low level driver to verify the settings.
629      */
630     if (port->ops->verify_port)
631         retval = port->ops->verify_port(port, &new_serial);
632
633     if ((new_serial.irq >= NR_IRQS) || (new_serial.irq < 0) ||
634         (new_serial.baud_base < 9600))
635         retval = -EINVAL;
636
637     if (retval)
638         goto exit;
639
640     if (change_port || change_irq) {
641         retval = -EBUSY;
642
643         /*
644          * Make sure that we are the sole user of this port.
645          */
646         if (state->count > 1)
647             goto exit;
648
649         /*
650          * We need to shutdown the serial port at the old
651          * port/type/irq combination.
652          */
653         uart_shutdown(info);
654     }
```

```
655
656     if (change_port) {
657         unsigned long old_iobase, old_mapbase;
658         unsigned int old_type, old_iotype, old_hub6, old_shift;
659
660         old_iobase = port->iobase;
661         old_mapbase = port->mapbase;
662         old_type = port->type;
663         old_hub6 = port->hub6;
664         old_iotype = port->iotype;
665         old_shift = port->regshift;
666
667         /*
668          * Free and release old regions
669          */
670         if (old_type != PORT_UNKNOWN)
671             port->ops->release_port(port);
672
673         port->iobase = new_port;
674         port->type = new_serial.type;
675         port->hub6 = new_serial.hub6;
676         port->iotype = new_serial.io_type;
677         port->regshift = new_serial.iomem_reg_shift;
678         port->mapbase = (unsigned long)new_serial.iomem_base;
679
680         /*
681          * Claim and map the new regions
682          */
683         if (port->type != PORT_UNKNOWN)
684             retval = port->ops->request_port(port);
685
686         /*
687          * If we fail to request resources for the
688          * new port, try to restore the old settings.
689          */
690         if (retval && old_type != PORT_UNKNOWN) {
691             port->iobase = old_iobase;
692             port->type = old_type;
693             port->hub6 = old_hub6;
694             port->iotype = old_iotype;
695             port->regshift = old_shift;
696             port->mapbase = old_mapbase;
697             retval = port->ops->request_port(port);
698             /*
699              * If we failed to restore the old settings,
700              * we fail like this.
701              */
```

```
702             if (retval)
703                 port->type = PORT_UNKNOWN;
704
705             /*
706              * We failed anyway.
707              */
708             retval = -EBUSY;
709         }
710     }
711
712     port->irq  = new_serial.irq;
713     port->uartclk = new_serial.baud_base * 16;
714     port->flags = ((port->flags & ~ASYNC_FLAGS) |
715             (new_serial.flags & ASYNC_FLAGS));
716     info->flags = ((port->flags & ~ASYNC_INTERNAL_FLAGS) |
717                (info->flags & ASYNC_INTERNAL_FLAGS));
718     state->custom_divisor = new_serial.custom_divisor;
719     state->close_delay = new_serial.close_delay * HZ / 100;
720     state->closing_wait = new_serial.closing_wait * HZ / 100;
721     info->tty->low_latency = (info->flags & ASYNC_LOW_LATENCY) ? 1 : 0;
722     port->fifosize = new_serial.xmit_fifo_size;
723
724   check_and_exit:
725     retval = 0;
726     if (port->type == PORT_UNKNOWN)
727         goto exit;
728     if (info->flags & ASYNC_INITIALIZED) {
729         if (((old_flags & info->flags) & ASYNC_SPD_MASK) ||
730             old_custom_divisor != state->custom_divisor) {
731             uart_update_altspeed(info);
732             uart_change_speed(info, NULL);
733         }
734     } else
735         retval = uart_startup(info);
736   exit:
737     up(&state->count_sem);
738     up(&port_sem);
739     return retval;
740   }
741
742
743   /*
744    * uart_get_lsr_info - get line status register info
745    */
746   static int uart_get_lsr_info(struct uart_info *info, unsigned int *value)
747   {
748     u_int result;
```

```
749     unsigned long flags;
750
751     spin_lock_irqsave(&info->lock, flags);
752     result = info->ops->tx_empty(info->port);
753     spin_unlock_irqrestore(&info->lock, flags);
754
755     /*
756      * If we're about to load something into the transmit
757      * register, we'll pretend the transmitter isn't empty to
758      * avoid a race condition (depending on when the transmit
759      * interrupt happens).
760      */
761     if (info->port->x_char ||
762         ((CIRC_CNT(info->xmit.head, info->xmit.tail,
763                UART_XMIT_SIZE) > 0) &&
764          !info->tty->stopped && !info->tty->hw_stopped))
765         result &= ~TIOCSER_TEMT;
766
767     return put_user(result, value);
768   }
769
770   static int uart_get_modem_info(struct uart_info *info, unsigned int *value)
771   {
772     unsigned int result = info->mctrl;
773
774     result |= info->ops->get_mctrl(info->port);
775
776     return put_user(result, value);
777   }
778
779   static int uart_set_modem_info(struct uart_info *info, unsigned int cmd,
780                  unsigned int *value)
781   {
782     unsigned int arg, old;
783     int ret = 0;
784
785     if (get_user(arg, value))
786         return -EFAULT;
787
788     spin_lock_irq(&info->lock);
789     old = info->mctrl;
790     switch (cmd) {
791     case TIOCMBIS:  info->mctrl |= arg; break;
792     case TIOCMBIC:  info->mctrl &= ~arg;    break;
793     case TIOCMSET:  info->mctrl = arg;  break;
794     default:    ret = -EINVAL;      break;
795     }
```

```
796     if (old != info->mctrl)
797         info->ops->set_mctrl(info->port, info->mctrl);
798     spin_unlock_irq(&info->lock);
799     return ret;
800   }
801
802   static void uart_break_ctl(struct tty_struct *tty, int break_state)
803   {
804     struct uart_info *info = tty->driver_data;
805     unsigned long flags;
806
807     if (info->port->type != PORT_UNKNOWN) {
808         spin_lock_irqsave(&info->lock, flags);
809         info->ops->break_ctl(info->port, break_state);
810         spin_unlock_irqrestore(&info->lock, flags);
811     }
812   }
813
814   static int uart_do_autoconfig(struct uart_info *info)
815   {
816     struct uart_port *port = info->port;
817     int flags, ret;
818
819     if (!capable(CAP_SYS_ADMIN))
820         return -EPERM;
821
822     /*
823      * Take the 'count' lock.  This prevents count
824      * from incrementing, and hence any extra opens
825      * of the port while we're auto-configging.
826      */
827     down(&info->state->count_sem);
828
829     ret = -EBUSY;
830     if (info->state->count == 1) {
831         uart_shutdown(info);
832
833         /*
834          * If we already have a port type configured,
835          * we must release its resources.
836          */
837         if (port->type != PORT_UNKNOWN)
838             port->ops->release_port(port);
839
840         flags = UART_CONFIG_TYPE;
841         if (port->flags & ASYNC_AUTO_IRQ)
842             flags |= UART_CONFIG_IRQ;
```

```
843
844         /*
845          * This will claim the ports resources if
846          * a port is found.
847          */
848         port->ops->config_port(port, flags);
849
850         ret = uart_startup(info);
851     }
852     up(&info->state->count_sem);
853     return ret;
854 }
855
856 /*
857  * Called from userspace.  We can use spin_lock_irq() here.
858  */
859 static int uart_ioctl(struct tty_struct *tty, struct file *file,
860                 unsigned int cmd, unsigned long arg)
861 {
862     struct uart_info *info = tty->driver_data;
863     struct uart_icount cprev, cnow;
864     struct serial_icounter_struct icount;
865     int ret = -ENOIOCTLCMD;
866
867     if ((cmd != TIOCGSERIAL) && (cmd != TIOCSSERIAL) &&
868         (cmd != TIOCSERCONFIG) && (cmd != TIOCSERGSTRUCT) &&
869         (cmd != TIOCMIWAIT) && (cmd != TIOCGICOUNT)) {
870         if (tty->flags & (1 << TTY_IO_ERROR))
871             return -EIO;
872     }
873
874     switch (cmd) {
875         case TIOCMGET:
876             ret = uart_get_modem_info(info, (unsigned int *)arg);
877             break;
878
879         case TIOCMBIS:
880         case TIOCMBIC:
881         case TIOCMSET:
882             ret = uart_set_modem_info(info, cmd,
883                                 (unsigned int *)arg);
884             break;
885
886         case TIOCGSERIAL:
887             ret = uart_get_info(info, (struct serial_struct *)arg);
888             break;
889
```

```
890         case TIOCSSERIAL:
891             ret = uart_set_info(info, (struct serial_struct *)arg);
892             break;
893 
894         case TIOCSERCONFIG:
895             ret = uart_do_autoconfig(info);
896             break;
897 
898         case TIOCSERGETLSR: /* Get line status register */
899             ret = uart_get_lsr_info(info, (unsigned int *)arg);
900             break;
901 
902         /*
903          * Wait for any of the 4 modem inputs (DCD,RI,DSR,CTS) to change
904          * - mask passed in arg for lines of interest
905          *   (use |'ed TIOCM_RNG/DSR/CD/CTS for masking)
906          * Caller should use TIOCGICOUNT to see which one it was
907          */
908         case TIOCMIWAIT:
909             spin_lock_irq(&info->lock);
910             /* note the counters on entry */
911             cprev = info->port->icount;
912             /* Force modem status interrupts on */
913             info->ops->enable_ms(info->port);
914             spin_unlock_irq(&info->lock);
915             while (1) {
916                 interruptible_sleep_on(&info->delta_msr_wait);
917                 /* see if a signal did it */
918                 if (signal_pending(current)) {
919                     ret = -ERESTARTSYS;
920                     break;
921                 }
922                 spin_lock_irq(&info->lock);
923                 cnow = info->port->icount; /* atomic copy */
924                 spin_unlock_irq(&info->lock);
925                 if (cnow.rng == cprev.rng && cnow.dsr == cprev.dsr &&
926                     cnow.dcd == cprev.dcd && cnow.cts == cprev.cts) {
927                         ret = -EIO; /* no change => error */
928                         break;
929                 }
930                 if (((arg & TIOCM_RNG) && (cnow.rng != cprev.rng)) ||
931                     ((arg & TIOCM_DSR) && (cnow.dsr != cprev.dsr)) ||
932                     ((arg & TIOCM_CD)  && (cnow.dcd != cprev.dcd)) ||
933                     ((arg & TIOCM_CTS) && (cnow.cts != cprev.cts))) {
934                         ret = 0;
935                         break;
936                 }
```

```
937                 cprev = cnow;
938             }
939             break;
940 
941         /*
942          * Get counter of input serial line interrupts (DCD,RI,DSR,CTS)
943          * Return: write counters to the user passed counter struct
944          * NB: both 1->0 and 0->1 transitions are counted except for
945          *     RI where only 0->1 is counted.
946          */
947         case TIOCGICOUNT:
948             spin_lock_irq(&info->lock);
949             cnow = info->port->icount;
950             spin_unlock_irq(&info->lock);
951 
952             icount.cts         = cnow.cts;
953             icount.dsr         = cnow.dsr;
954             icount.rng         = cnow.rng;
955             icount.dcd         = cnow.dcd;
956             icount.rx          = cnow.rx;
957             icount.tx          = cnow.tx;
958             icount.frame       = cnow.frame;
959             icount.overrun     = cnow.overrun;
960             icount.parity      = cnow.parity;
961             icount.brk         = cnow.brk;
962             icount.buf_overrun = cnow.buf_overrun;
963 
964             ret = copy_to_user((void *)arg, &icount, sizeof(icount))
965                     ? -EFAULT : 0;
966             break;
967 
968         case TIOCSERGWILD: /* obsolete */
969         case TIOCSERSWILD: /* obsolete */
970             ret = 0;
971             break;
972 
973         default:
974             if (info->ops->ioctl)
975                 ret = info->ops->ioctl(info->port, cmd, arg);
976             break;
977     }
978     return ret;
979   }
980 
981      static  void  uart_set_termios(struct  tty_struct  *tty,  struct  termios
*old_termios)
982   {
```

```
983     struct uart_info *info = tty->driver_data;
984     unsigned long flags;
985     unsigned int cflag = tty->termios->c_cflag;
986
987     if ((cflag ^ old_termios->c_cflag) == 0 &&
988         RELEVENT_IFLAG(tty->termios->c_iflag ^ old_termios->c_iflag) == 0)
989         return;
990
991     uart_change_speed(info, old_termios);
992
993     spin_lock_irqsave(&info->lock, flags);
994
995     /* Handle transition to B0 status */
996     if ((old_termios->c_cflag & CBAUD) && !(cflag & CBAUD)) {
997         info->mctrl &= ~(TIOCM_RTS | TIOCM_DTR);
998         info->ops->set_mctrl(info->port, info->mctrl);
999     }
1000
1001    /* Handle transition away from B0 status */
1002    if (!(old_termios->c_cflag & CBAUD) && (cflag & CBAUD)) {
1003        info->mctrl |= TIOCM_DTR;
1004        if (!(cflag & CRTSCTS) ||
1005            !test_bit(TTY_THROTTLED, &tty->flags))
1006            info->mctrl |= TIOCM_RTS;
1007        info->ops->set_mctrl(info->port, info->mctrl);
1008    }
1009
1010    /* Handle turning off CRTSCTS */
1011    if ((old_termios->c_cflag & CRTSCTS) && !(cflag & CRTSCTS)) {
1012        tty->hw_stopped = 0;
1013        __uart_start(tty);
1014    }
1015    spin_unlock_irqrestore(&info->lock, flags);
1016
1017   #if 0
1018    /*
1019     * No need to wake up processes in open wait, since they
1020     * sample the CLOCAL flag once, and don't recheck it.
1021     * XXX  It's not clear whether the current behavior is correct
1022     * or not.  Hence, this may change.....
1023     */
1024    if (!(old_termios->c_cflag & CLOCAL) &&
1025        (tty->termios->c_cflag & CLOCAL))
1026        wake_up_interruptible(&info->open_wait);
1027   #endif
1028   }
1029
```

```
1030  /*
1031   * In 2.4.5, calls to this will be serialized via the BKL in
1032   *  linux/drivers/char/tty_io.c:tty_release()
1033   *  linux/drivers/char/tty_io.c:do_tty_handup()
1034   */
1035  static void uart_close(struct tty_struct *tty, struct file *filp)
1036  {
1037   struct uart_driver *drv = (struct uart_driver *)tty->driver.driver_state;
1038   struct uart_info *info = tty->driver_data;
1039   struct uart_state *state;
1040   unsigned long flags;
1041
1042   if (!info)
1043       return;
1044
1045   state = info->state;
1046
1047  #ifdef DEBUG
1048   printk("uart_close() called\n");
1049  #endif
1050
1051   /*
1052    * This is safe, as long as the BKL exists in
1053    * do_tty_hangup(), and we're protected by the BKL.
1054    */
1055   if (tty_hung_up_p(filp))
1056       goto done;
1057
1058   down(&state->count_sem);
1059   spin_lock_irqsave(&info->lock, flags);
1060   if ((tty->count == 1) && (state->count != 1)) {
1061       /*
1062        * Uh, oh.  tty->count is 1, which means that the tty
1063        * structure will be freed.  state->count should always
1064        * be one in these conditions.  If it's greater than
1065        * one, we've got real problems, since it means the
1066        * serial port won't be shutdown.
1067        */
1068       printk("uart_close: bad serial port count; tty->count is 1, "
1069              "state->count is %d\n", state->count);
1070       state->count = 1;
1071   }
1072   if (--state->count < 0) {
1073       printk("rs_close: bad serial port count for %s%d: %d\n",
1074              tty->driver.name, info->port->line, state->count);
1075       state->count = 0;
1076   }
```

```
1077    if (state->count) {
1078        spin_unlock_irqrestore(&info->lock, flags);
1079        up(&state->count_sem);
1080        goto done;
1081    }
1082    info->flags |= ASYNC_CLOSING;
1083    spin_unlock_irqrestore(&info->lock, flags);
1084    up(&state->count_sem);
1085
1086    /*
1087     * Save the termios structure, since this port may have
1088     * separate termios for callout and dialin.
1089     */
1090    if (info->flags & ASYNC_NORMAL_ACTIVE)
1091        info->state->normal_termios = *tty->termios;
1092    if (info->flags & ASYNC_CALLOUT_ACTIVE)
1093        info->state->callout_termios = *tty->termios;
1094    /*
1095     * Now we wait for the transmit buffer to clear; and we notify
1096     * the line discipline to only process XON/XOFF characters.
1097     */
1098    tty->closing = 1;
1099    if (info->state->closing_wait != ASYNC_CLOSING_WAIT_NONE)
1100        tty_wait_until_sent(tty, info->state->closing_wait);
1101    /*
1102     * At this point, we stop accepting input.  To do this, we
1103     * disable the receive line status interrupts.
1104     */
1105    if (info->flags & ASYNC_INITIALIZED) {
1106        info->ops->stop_rx(info->port);
1107        /*
1108         * Before we drop DTR, make sure the UART transmitter
1109         * has completely drained; this is especially
1110         * important if there is a transmit FIFO!
1111         */
1112        uart_wait_until_sent(tty, info->timeout);
1113    }
1114    uart_shutdown(info);
1115    if (tty->driver.flush_buffer)
1116        tty->driver.flush_buffer(tty);
1117    if (tty->ldisc.flush_buffer)
1118        tty->ldisc.flush_buffer(tty);
1119    tty->closing = 0;
1120    info->event = 0;
1121    info->tty = NULL;
1122    if (info->blocked_open) {
1123        if (info->state->close_delay) {
```

```
1124            set_current_state(TASK_INTERRUPTIBLE);
1125            schedule_timeout(info->state->close_delay);
1126            set_current_state(TASK_RUNNING);
1127        }
1128        wake_up_interruptible(&info->open_wait);
1129    } else {
1130    #ifdef CONFIG_PM
1131        /*
1132         * Put device into D3 state.
1133         */
1134        pm_send(info->state->pm, PM_SUSPEND, (void *)3);
1135    #else
1136        if (info->ops->pm)
1137            info->ops->pm(info->port, 3, 0);
1138    #endif
1139    }
1140
1141    info->flags &= ~(ASYNC_NORMAL_ACTIVE|ASYNC_CALLOUT_ACTIVE|
1142             ASYNC_CLOSING);
1143    wake_up_interruptible(&info->close_wait);
1144
1145    done:
1146    if (drv->owner)
1147        __MOD_DEC_USE_COUNT(drv->owner);
1148    }
1149
1150    static void uart_wait_until_sent(struct tty_struct *tty, int timeout)
1151    {
1152    struct uart_info *info = (struct uart_info *) tty->driver_data;
1153    unsigned long char_time, expire;
1154
1155    if (info->port->type == PORT_UNKNOWN ||
1156        info->port->fifosize == 0)
1157        return;
1158
1159    /*
1160     * Set the check interval to be 1/5 of the estimated time to
1161     * send a single character, and make it at least 1.  The check
1162     * interval should also be less than the timeout.
1163     *
1164     * Note: we have to use pretty tight timings here to satisfy
1165     * the NIST-PCTS.
1166     */
1167    char_time = (info->timeout - HZ/50) / info->port->fifosize;
1168    char_time = char_time / 5;
1169    if (char_time == 0)
1170        char_time = 1;
```

```
1171    if (timeout && timeout < char_time)
1172        char_time = timeout;
1173    /*
1174     * If the transmitter hasn't cleared in twice the approximate
1175     * amount of time to send the entire FIFO, it probably won't
1176     * ever clear.  This assumes the UART isn't doing flow
1177     * control, which is currently the case.  Hence, if it ever
1178     * takes longer than info->timeout, this is probably due to a
1179     * UART bug of some kind.  So, we clamp the timeout parameter at
1180     * 2*info->timeout.
1181     */
1182    if (!timeout || timeout > 2 * info->timeout)
1183        timeout = 2 * info->timeout;
1184
1185    expire = jiffies + timeout;
1186   #ifdef DEBUG
1187    printk("uart_wait_until_sent(%d), jiff=%lu, expire=%lu...\n",
1188           MINOR(tty->device) - tty->driver.minor_start, jiffies,
1189           expire);
1190   #endif
1191    while (!info->ops->tx_empty(info->port)) {
1192        set_current_state(TASK_INTERRUPTIBLE);
1193        schedule_timeout(char_time);
1194        if (signal_pending(current))
1195            break;
1196        if (timeout && time_after(jiffies, expire))
1197            break;
1198    }
1199    set_current_state(TASK_RUNNING); /* might not be needed */
1200   }
1201
1202   /*
1203    * This is called with the BKL in effect
1204    *  linux/drivers/char/tty_io.c:do_tty_hangup()
1205    */
1206   static void uart_hangup(struct tty_struct *tty)
1207   {
1208    struct uart_info *info = tty->driver_data;
1209    struct uart_state *state = info->state;
1210
1211    uart_flush_buffer(tty);
1212    if (info->flags & ASYNC_CLOSING)
1213        return;
1214    uart_shutdown(info);
1215    info->event = 0;
1216    state->count = 0;
1217    info->flags &= ~(ASYNC_NORMAL_ACTIVE|ASYNC_CALLOUT_ACTIVE);
```

```
1218    info->tty = NULL;
1219    wake_up_interruptible(&info->open_wait);
1220   }
1221
1222   static int uart_block_til_ready(struct tty_struct *tty, struct file *filp,
1223                struct uart_info *info)
1224   {
1225    DECLARE_WAITQUEUE(wait, current);
1226    struct uart_state *state = info->state;
1227    unsigned long flags;
1228    int do_clocal = 0, extra_count = 0, retval;
1229
1230    /*
1231     * If the device is in the middle of being closed, then block
1232     * until it's done, and then try again.
1233     */
1234    if (tty_hung_up_p(filp) ||
1235        (info->flags & ASYNC_CLOSING)) {
1236        if (info->flags & ASYNC_CLOSING)
1237            interruptible_sleep_on(&info->close_wait);
1238        return (info->flags & ASYNC_HUP_NOTIFY) ?
1239            -EAGAIN : -ERESTARTSYS;
1240    }
1241
1242    /*
1243     * If this is a callout device, then just make sure the normal
1244     * device isn't being used.
1245     */
1246    if (tty->driver.subtype == SERIAL_TYPE_CALLOUT) {
1247        if (info->flags & ASYNC_NORMAL_ACTIVE)
1248            return -EBUSY;
1249        if ((info->flags & ASYNC_CALLOUT_ACTIVE) &&
1250            (info->flags & ASYNC_SESSION_LOCKOUT) &&
1251            (info->session != current->session))
1252            return -EBUSY;
1253        if ((info->flags & ASYNC_CALLOUT_ACTIVE) &&
1254            (info->flags & ASYNC_PGRP_LOCKOUT) &&
1255            (info->pgrp != current->pgrp))
1256            return -EBUSY;
1257        info->flags |= ASYNC_CALLOUT_ACTIVE;
1258        return 0;
1259    }
1260
1261    /*
1262     * If non-blocking mode is set, or the port is not enabled,
1263     * then make the check up front and then exit.  Note that
1264     * we have set TTY_IO_ERROR for a non-enabled port.
```

```
1265     */
1266    if ((filp->f_flags & O_NONBLOCK) ||
1267        (tty->flags & (1 << TTY_IO_ERROR))) {
1268        if (info->flags & ASYNC_CALLOUT_ACTIVE)
1269            return -EBUSY;
1270        info->flags |= ASYNC_NORMAL_ACTIVE;
1271        return 0;
1272    }
1273
1274    if (info->flags & ASYNC_CALLOUT_ACTIVE) {
1275        if (state->normal_termios.c_cflag & CLOCAL)
1276            do_clocal = 1;
1277    } else {
1278        if (tty->termios->c_cflag & CLOCAL)
1279            do_clocal = 1;
1280    }
1281
1282    /*
1283     * Block waiting for the carrier detect and the line to become
1284     * free (i.e., not in use by the callout).  While we are in
1285     * this loop, state->count is dropped by one, so that
1286     * rs_close() knows when to free things.  We restore it upon
1287     * exit, either normal or abnormal.
1288     */
1289    retval = 0;
1290    add_wait_queue(&info->open_wait, &wait);
1291    down(&state->count_sem);
1292    spin_lock_irqsave(&info->lock, flags);
1293    if (!tty_hung_up_p(filp)) {
1294        extra_count = 1;
1295        state->count--;
1296    }
1297    spin_unlock_irqrestore(&info->lock, flags);
1298    info->blocked_open++;
1299    up(&state->count_sem);
1300    while (1) {
1301        spin_lock_irqsave(&info->lock, flags);
1302        if (!(info->flags & ASYNC_CALLOUT_ACTIVE) &&
1303            (tty->termios->c_cflag & CBAUD)) {
1304            info->mctrl |= TIOCM_DTR | TIOCM_RTS;
1305            info->ops->set_mctrl(info->port, info->mctrl);
1306        }
1307        spin_unlock_irqrestore(&info->lock, flags);
1308        set_current_state(TASK_INTERRUPTIBLE);
1309        if (tty_hung_up_p(filp) ||
1310            !(info->flags & ASYNC_INITIALIZED)) {
1311            if (info->flags & ASYNC_HUP_NOTIFY)
```

```
1312                retval = -EAGAIN;
1313            else
1314                retval = -ERESTARTSYS;
1315            break;
1316        }
1317        if (!(info->flags & ASYNC_CALLOUT_ACTIVE) &&
1318            !(info->flags & ASYNC_CLOSING) &&
1319            (do_clocal ||
1320             (info->ops->get_mctrl(info->port) & TIOCM_CAR)))
1321            break;
1322        if (signal_pending(current)) {
1323            retval = -ERESTARTSYS;
1324            break;
1325        }
1326        schedule();
1327    }
1328    set_current_state(TASK_RUNNING);
1329    remove_wait_queue(&info->open_wait, &wait);
1330    down(&state->count_sem);
1331    if (extra_count)
1332        state->count++;
1333    info->blocked_open--;
1334    up(&state->count_sem);
1335    if (retval)
1336        return retval;
1337    info->flags |= ASYNC_NORMAL_ACTIVE;
1338    return 0;
1339   }
1340
1341   static struct uart_info *uart_get(struct uart_driver *drv, int line)
1342   {
1343    struct uart_state *state = drv->state + line;
1344    struct uart_info *info;
1345
1346    down(&state->count_sem);
1347    state->count++;
1348    if (state->info)
1349        goto out;
1350
1351    info = kmalloc(sizeof(struct uart_info), GFP_KERNEL);
1352    if (info) {
1353        memset(info, 0, sizeof(struct uart_info));
1354        init_waitqueue_head(&info->open_wait);
1355        init_waitqueue_head(&info->close_wait);
1356        init_waitqueue_head(&info->delta_msr_wait);
1357        info->port  = state->port;
1358        info->flags = info->port->flags;
```

```
1359        info->ops   = info->port->ops;
1360        info->state = state;
1361        tasklet_init(&info->tlet, uart_tasklet_action,
1362                 (unsigned long)info);
1363    }
1364    if (state->info)
1365        kfree(info);
1366    else
1367        state->info = info;
1368   out:
1369    up(&state->count_sem);
1370    return state->info;
1371   }
1372
1373   /*
1374    * Make sure we have the temporary buffer allocated.  Note
1375    * that we set retval appropriately above, and we rely on
1376    * this.
1377    */
1378   static inline int uart_alloc_tmpbuf(void)
1379   {
1380    if (!tmp_buf) {
1381        unsigned long buf = get_zeroed_page(GFP_KERNEL);
1382        if (!tmp_buf) {
1383            if (buf)
1384                tmp_buf = (u_char *)buf;
1385            else
1386                return -ENOMEM;
1387        } else
1388            free_page(buf);
1389    }
1390    return 0;
1391   }
1392
1393   /*
1394    * In 2.4.5, calls to uart_open are serialised by the BKL in
1395    *   linux/fs/devices.c:chrdev_open()
1396    * Note that if this fails, then uart_close() _will_ be called.
1397    */
1398   static int uart_open(struct tty_struct *tty, struct file *filp)
1399   {
1400    struct uart_driver *drv = (struct uart_driver *)tty->driver.driver_state;
1401    struct uart_info *info;
1402    int retval, line = MINOR(tty->device) - tty->driver.minor_start;
1403
1404   #ifdef DEBUG
1405    printk("uart_open(%d) called\n", line);
```

```
1406  #endif
1407
1408   retval = -ENODEV;
1409   if (line >= tty->driver.num)
1410       goto fail;
1411
1412   if (!try_inc_mod_count(drv->owner))
1413       goto fail;
1414
1415   info = uart_get(drv, line);
1416   retval = -ENOMEM;
1417   if (!info)
1418       goto out;
1419
1420   /*
1421    * Set the tty driver_data.  If we fail from this point on,
1422    * the generic tty layer will cause uart_close(), which will
1423    * decrement the module use count.
1424    */
1425   tty->driver_data = info;
1426   info->tty = tty;
1427   info->tty->low_latency = (info->flags & ASYNC_LOW_LATENCY) ? 1 : 0;
1428
1429   if (uart_alloc_tmpbuf())
1430       goto fail;
1431
1432   /*
1433    * If the port is in the middle of closing, bail out now.
1434    */
1435   if (tty_hung_up_p(filp) ||
1436       (info->flags & ASYNC_CLOSING)) {
1437       if (info->flags & ASYNC_CLOSING)
1438           interruptible_sleep_on(&info->close_wait);
1439       retval = (info->flags & ASYNC_HUP_NOTIFY) ?
1440           -EAGAIN : -ERESTARTSYS;
1441       goto fail;
1442   }
1443
1444   /*
1445    * Make sure the device is in D0 state.
1446    */
1447   if (info->state->count == 1)
1448  #ifdef CONFIG_PM
1449       pm_send(info->state->pm, PM_RESUME, (void *)0);
1450  #else
1451       if (info->ops->pm)
1452           info->ops->pm(info->port, 0, 3);
```

```
1453  #endif
1454
1455   /*
1456    * Start up the serial port
1457    */
1458   retval = uart_startup(info);
1459   if (retval)
1460       goto fail;
1461
1462   retval = uart_block_til_ready(tty, filp, info);
1463   if (retval)
1464       goto fail;
1465
1466   if (info->state->count == 1) {
1467       int changed_termios = 0;
1468
1469       if (info->flags & ASYNC_SPLIT_TERMIOS) {
1470           if (tty->driver.subtype == SERIAL_TYPE_NORMAL)
1471               *tty->termios = info->state->normal_termios;
1472           else
1473               *tty->termios = info->state->callout_termios;
1474           changed_termios = 1;
1475       }
1476
1477  //#ifdef CONFIG_SERIAL_CORE_CONSOLE
1478       /*
1479        * Copy across the serial console cflag setting
1480        */
1481       {
1482           struct console *c = drv->cons;
1483           if (c && c->cflag && c->index == line) {
1484               tty->termios->c_cflag = c->cflag;
1485               c->cflag = 0;
1486               changed_termios = 1;
1487           }
1488       }
1489  //#endif
1490       if (changed_termios)
1491           uart_change_speed(info, NULL);
1492   }
1493
1494   info->session = current->session;
1495   info->pgrp = current->pgrp;
1496   return 0;
1497
1498  out:
1499   if (drv->owner)
```

```
1500        __MOD_DEC_USE_COUNT(drv->owner);
1501    fail:
1502     return retval;
1503    }
1504
1505    #ifdef CONFIG_PROC_FS
1506
1507    static const char *uart_type(struct uart_port *port)
1508    {
1509     const char *str = NULL;
1510
1511     if (port->ops->type)
1512         str = port->ops->type(port);
1513
1514     if (!str)
1515         str = "unknown";
1516
1517     return str;
1518    }
1519
1520    static int uart_line_info(char *buf, struct uart_driver *drv, int i)
1521    {
1522     struct uart_state *state = drv->state + i;
1523     struct uart_port *port = state->port;
1524     char stat_buf[32];
1525     u_int status;
1526     int ret;
1527
1528     ret = sprintf(buf, "%d: uart:%s port:%08X irq:%d",
1529             port->line, uart_type(port),
1530             port->iobase, port->irq);
1531
1532     if (port->type == PORT_UNKNOWN) {
1533         strcat(buf, "\n");
1534         return ret + 1;
1535     }
1536
1537     status = port->ops->get_mctrl(port);
1538
1539     ret += sprintf(buf + ret, " tx:%d rx:%d",
1540             port->icount.tx, port->icount.rx);
1541     if (port->icount.frame)
1542         ret += sprintf(buf + ret, " fe:%d",
1543             port->icount.frame);
1544     if (port->icount.parity)
1545         ret += sprintf(buf + ret, " pe:%d",
1546             port->icount.parity);
```

```
1547    if (port->icount.brk)
1548        ret += sprintf(buf + ret, " brk:%d",
1549            port->icount.brk);
1550    if (port->icount.overrun)
1551        ret += sprintf(buf + ret, " oe:%d",
1552            port->icount.overrun);
1553
1554   #define INFOBIT(bit,str) \
1555    if (state->info && state->info->mctrl & (bit)) \
1556        strcat(stat_buf, (str))
1557   #define STATBIT(bit,str) \
1558    if (status & (bit)) \
1559        strcat(stat_buf, (str))
1560
1561    stat_buf[0] = '\0';
1562    stat_buf[1] = '\0';
1563    INFOBIT(TIOCM_RTS, "|RTS");
1564    STATBIT(TIOCM_CTS, "|CTS");
1565    INFOBIT(TIOCM_DTR, "|DTR");
1566    STATBIT(TIOCM_DSR, "|DSR");
1567    STATBIT(TIOCM_CAR, "|CD");
1568    STATBIT(TIOCM_RNG, "|RI");
1569    if (stat_buf[0])
1570        stat_buf[0] = ' ';
1571    strcat(stat_buf, "\n");
1572
1573    ret += sprintf(buf + ret, stat_buf);
1574    return ret;
1575   }
1576
1577   static int uart_read_proc(char *page, char **start, off_t off,
1578                int count, int *eof, void *data)
1579   {
1580    struct tty_driver *ttydrv = data;
1581    struct uart_driver *drv = ttydrv->driver_state;
1582    int i, len = 0, l;
1583    off_t begin = 0;
1584
1585    len += sprintf(page, "serinfo:1.0 driver%s%s revision:%s\n",
1586            "", "", "");
1587    for (i = 0; i < drv->nr && len < PAGE_SIZE - 96; i++) {
1588        l = uart_line_info(page + len, drv, i);
1589        len += l;
1590        if (len + begin > off + count)
1591            goto done;
1592        if (len + begin < off) {
1593            begin += len;
```

```
1594            len = 0;
1595        }
1596    }
1597    *eof = 1;
1598   done:
1599    if (off >= len + begin)
1600        return 0;
1601    *start = page + (off - begin);
1602    return (count < begin + len - off) ? count : (begin + len - off);
1603   }
1604   #endif
1605
1606   //#ifdef CONFIG_SERIAL_CORE_CONSOLE
1607   /*
1608    *   Check whether an invalid uart number has been specified, and
1609    *   if so, search for the first available port that does have
1610    *   console support.
1611    */
1612   struct uart_port * __init
1613   uart_get_console(struct uart_port *ports, int nr, struct console *co)
1614   {
1615    int idx = co->index;
1616
1617    if (idx < 0 || idx >= nr || (ports[idx].iobase == 0 &&
1618                        ports[idx].membase == NULL))
1619        for (idx = 0; idx < nr; idx++)
1620            if (ports[idx].iobase != 0 ||
1621                ports[idx].membase != NULL)
1622                break;
1623
1624    co->index = idx;
1625
1626    return ports + idx;
1627   }
1628
1629   /**
1630    *   uart_parse_options - Parse serial port baud/parity/bits/flow contro.
1631    *   @options: pointer to option string
1632    *   @baud: pointer to an 'int' variable for the baud rate.
1633    *   @parity: pointer to an 'int' variable for the parity.
1634    *   @bits: pointer to an 'int' variable for the number of data bits.
1635    *   @flow: pointer to an 'int' variable for the flow control character.
1636    *
1637    *   uart_parse_options decodes a string containing the serial console
1638    *   options.  The format of the string is <baud><parity><bits><flow>,
1639    *   eg: 115200n8r
1640    */
```

```
1641  void __init
1642  uart_parse_options(char *options, int *baud, int *parity, int *bits, int *flow)
1643  {
1644   char *s = options;
1645
1646   *baud = simple_strtoul(s, NULL, 10);
1647   while (*s >= '0' && *s <= '9')
1648       s++;
1649   if (*s)
1650       *parity = *s++;
1651   if (*s)
1652       *bits = *s++ - '0';
1653   if (*s)
1654       *flow = *s;
1655  }
1656
1657  /**
1658   *   uart_set_options - setup the serial console parameters
1659   *   @port: pointer to the serial ports uart_port structure
1660   *   @co: console pointer
1661   *   @baud: baud rate
1662   *   @parity: parity character - 'n' (none), 'o' (odd), 'e' (even)
1663   *   @bits: number of data bits
1664   *   @flow: flow control character - 'r' (rts)
1665   */
1666  int __init
1667  uart_set_options(struct uart_port *port, struct console *co,
1668        int baud, int parity, int bits, int flow)
1669  {
1670   u_int cflag = CREAD | HUPCL | CLOCAL;
1671   u_int quot;
1672
1673   /*
1674    * Construct a cflag setting.
1675    */
1676   switch (baud) {
1677   case 1200:  cflag |= B1200;         break;
1678   case 2400:  cflag |= B2400;         break;
1679   case 4800:  cflag |= B4800;         break;
1680   case 9600:  cflag |= B9600;         break;
1681   case 19200: cflag |= B19200;        break;
1682  //   default:    cflag |= B38400;  baud = 38400; break;
1683   default:    cflag |= B115200;  baud = 115200;   break;
1684   case 57600: cflag |= B57600;        break;
1685   case 115200:    cflag |= B115200;       break;
1686   case 230400:    cflag |= B230400;       break;
1687   case 460800:    cflag |= B460800;       break;
```

```
1688    }
1689
1690    if (bits == 7)
1691        cflag |= CS7;
1692    else
1693        cflag |= CS8;
1694
1695    switch (parity) {
1696    case 'o': case 'O':
1697        cflag |= PARODD;
1698        /*fall through*/
1699    case 'e': case 'E':
1700        cflag |= PARENB;
1701        break;
1702    }
1703
1704    co->cflag = cflag;
1705    quot = (port->uartclk / (16 * baud));
1706    port->ops->change_speed(port, cflag, 0, quot);
1707
1708    return 0;
1709   }
1710
1711   extern void ambauart_console_init(void);
1712   extern void anakin_console_init(void);
1713   extern void clps711xuart_console_init(void);
1714   extern void rs285_console_init(void);
1715   extern void sa1100_rs_console_init(void);
1716   extern void serial8250_console_init(void);
1717   extern void samsung_console_init(void);
1718
1719   /*
1720    * Central "initialise all serial consoles" container.  Needs to be killed.
1721    */
1722   void __init uart_console_init(void)
1723   {
1724   #ifdef CONFIG_SERIAL_AMBA_CONSOLE
1725    ambauart_console_init();
1726   #endif
1727   #ifdef CONFIG_SERIAL_ANAKIN_CONSOLE
1728    anakin_console_init();
1729   #endif
1730   #ifdef CONFIG_SERIAL_CLPS711X_CONSOLE
1731    clps711xuart_console_init();
1732   #endif
1733   #ifdef CONFIG_SERIAL_21285_CONSOLE
1734    rs285_console_init();
```

```
1735   #endif
1736   #ifdef CONFIG_SERIAL_SA1100_CONSOLE
1737    sa1100_rs_console_init();
1738   #endif
1739   #ifdef CONFIG_SERIAL_8250_CONSOLE
1740    serial8250_console_init();
1741   #endif
1742   #ifdef CONFIG_SERIAL_UART00_CONSOLE
1743    uart00_console_init();
1744   #endif
1745   #ifdef CONFIG_SERIAL_SAMSUNG_CONSOLE
1746    samsung_console_init();
1747   #endif
1748   #ifdef CONFIG_SERIAL_S3C44B0_CONSOLE
1749    samsung_console_init();
1750   #endif
1751   }
1752   //#endif /* CONFIG_SERIAL_CORE_CONSOLE */
1753
1754   #ifdef CONFIG_PM
1755   /*
1756    *  Serial port power management.
1757    *
1758    * This is pretty coarse at the moment - either all on or all off.  We
1759    * should probably some day do finer power management here some day.
1760    *
1761    * We don't actually save any state; the serial driver has enough
1762    * state held internally to re-setup the port when we come out of D3.
1763    */
1764    static int uart_pm_set_state(struct uart_state *state, int pm_state, int
oldstate)
1765   {
1766    struct uart_port *port = state->port;
1767    struct uart_ops *ops = port->ops;
1768    int running = state->info &&
1769              state->info->flags & ASYNC_INITIALIZED;
1770
1771    if (port->type == PORT_UNKNOWN)
1772        return 0;
1773
1774   //printk("pm: %08x: %d -> %d, %srunning\n", port->iobase, dev->state, pm_state,
running ? "" : "not ");
1775    if (pm_state == 0) {
1776        if (ops->pm)
1777            ops->pm(port, pm_state, oldstate);
1778        if (running) {
1779            ops->set_mctrl(port, 0);
```

```
1780            ops->startup(port, state->info);
1781            uart_change_speed(state->info, NULL);
1782            ops->set_mctrl(port, state->info->mctrl);
1783            ops->start_tx(port, 1, 0);
1784        }
1785
1786        /*
1787         * Re-enable the console device after suspending.
1788         */
1789        if (state->cons && state->cons->index == port->line)
1790            state->cons->flags |= CON_ENABLED;
1791    } else if (pm_state == 1) {
1792        if (ops->pm)
1793            ops->pm(port, pm_state, oldstate);
1794    } else {
1795        /*
1796         * Disable the console device before suspending.
1797         */
1798        if (state->cons && state->cons->index == port->line)
1799            state->cons->flags &= ~CON_ENABLED;
1800
1801        if (running) {
1802            ops->stop_tx(port, 0);
1803            ops->set_mctrl(port, 0);
1804            ops->stop_rx(port);
1805            ops->shutdown(port, state->info);
1806        }
1807        if (ops->pm)
1808            ops->pm(port, pm_state, oldstate);
1809    }
1810    return 0;
1811  }
1812
1813  /*
1814   *  Wakeup support.
1815   */
1816  static int uart_pm_set_wakeup(struct uart_state *state, int data)
1817  {
1818    int err = 0;
1819
1820    if (state->port->ops->set_wake)
1821        err = state->port->ops->set_wake(state->port, data);
1822
1823    return err;
1824  }
1825
1826  static int uart_pm(struct pm_dev *dev, pm_request_t rqst, void *data)
```

```
1827  {
1828   struct uart_state *state = dev->data;
1829   int err = 0;
1830
1831   switch (rqst) {
1832   case PM_SUSPEND:
1833   case PM_RESUME:
1834       err = uart_pm_set_state(state, (int)data, dev->state);
1835       break;
1836
1837   case PM_SET_WAKEUP:
1838       err = uart_pm_set_wakeup(state, (int)data);
1839       break;
1840   }
1841   return err;
1842  }
1843  #endif
1844
1845  static inline void
1846  uart_report_port(struct uart_driver *drv, struct uart_port *port)
1847  {
1848   printk("%s%d at ", drv->normal_name, port->line);
1849   switch (port->iotype) {
1850   case SERIAL_IO_PORT:
1851       printk("I/O 0x%x", port->iobase);
1852       break;
1853   case SERIAL_IO_HUB6:
1854       printk("I/O 0x%x offset 0x%x", port->iobase, port->hub6);
1855       break;
1856   case SERIAL_IO_MEM:
1857       printk("MEM 0x%lx", port->mapbase);
1858       break;
1859   }
1860   printk(" (irq = %d) is a %s\n", port->irq, uart_type(port));
1861  }
1862
1863  static void
1864  uart_setup_port(struct uart_driver *drv, struct uart_state *state)
1865  {
1866   struct uart_port *port = state->port;
1867   int flags = UART_CONFIG_TYPE;
1868
1869   init_MUTEX(&state->count_sem);
1870
1871   state->close_delay  = 5 * HZ / 10;
1872   state->closing_wait = 30 * HZ;
1873
```

```
1874    port->type = PORT_UNKNOWN;
1875
1876   #ifdef CONFIG_PM
1877    state->cons = drv->cons;
1878    state->pm = pm_register(PM_SYS_DEV, PM_SYS_COM, uart_pm);
1879    if (state->pm)
1880        state->pm->data = state;
1881   #endif
1882
1883    /*
1884     * If there isn't a port here, don't do anything further.
1885     */
1886    if (!port->iobase && !port->mapbase)
1887        return;
1888
1889    /*
1890     * Now do the auto configuration stuff.  Note that config_port
1891     * is expected to claim the resources and map the port for us.
1892     */
1893    if (port->flags & ASYNC_AUTO_IRQ)
1894        flags |= UART_CONFIG_IRQ;
1895    if (port->flags & ASYNC_BOOT_AUTOCONF)
1896        port->ops->config_port(port, flags);
1897
1898    /*
1899     * Only register this port if it is detected.
1900     */
1901    if (port->type != PORT_UNKNOWN) {
1902        tty_register_devfs(drv->normal_driver, 0, drv->minor +
1903                    state->port->line);
1904        tty_register_devfs(drv->callout_driver, 0, drv->minor +
1905                    state->port->line);
1906        uart_report_port(drv, port);
1907    }
1908
1909   #ifdef CONFIG_PM
1910    /*
1911     * Power down all ports by default, except the console if we have one.
1912     */
1913    if (state->pm && (!drv->cons || port->line != drv->cons->index))
1914        pm_send(state->pm, PM_SUSPEND, (void *)3);
1915   #endif
1916   }
1917
1918   /*
1919    * Register a set of ports with the core driver.  Note that we don't
1920    * printk any information about the ports; that is up to the low level
```

```
1921    * driver to do if they so wish.
1922    */
1923   int uart_register_driver(struct uart_driver *drv)
1924   {
1925    struct tty_driver *normal, *callout;
1926    int i, retval;
1927
1928    if (drv->state)
1929        panic("drv->state already allocated\n");
1930
1931    /*
1932     * Maybe we should be using a slab cache for this, especially if
1933     * we have a large number of ports to handle.  Note that we also
1934     * allocate space for an integer for reference counting.
1935     */
1936    drv->state = kmalloc(sizeof(struct uart_state) * drv->nr +
1937                 sizeof(int), GFP_KERNEL);
1938    retval = -ENOMEM;
1939    if (!drv->state)
1940        goto out;
1941
1942    memset(drv->state, 0, sizeof(struct uart_state) * drv->nr +
1943           sizeof(int));
1944
1945    normal  = drv->normal_driver;
1946    callout = drv->callout_driver;
1947
1948    normal->magic       = TTY_DRIVER_MAGIC;
1949    normal->driver_name = drv->normal_name;
1950    normal->name        = drv->normal_name;
1951    normal->major       = drv->normal_major;
1952    normal->minor_start = drv->minor;
1953    normal->num     = drv->nr;
1954    normal->type        = TTY_DRIVER_TYPE_SERIAL;
1955    normal->subtype     = SERIAL_TYPE_NORMAL;
1956    normal->init_termios    = tty_std_termios;
1957    normal->init_termios.c_cflag = B115200 | CS8 | CREAD | HUPCL | CLOCAL;
1958   //   normal->init_termios.c_cflag = B38400 | CS8 | CREAD | HUPCL | CLOCAL;
1959    normal->flags       = TTY_DRIVER_REAL_RAW | TTY_DRIVER_NO_DEVFS;
1960    normal->refcount    = (int *)(drv->state + drv->nr);
1961    normal->table       = drv->table;
1962    normal->termios     = drv->termios;
1963    normal->termios_locked  = drv->termios_locked;
1964    normal->driver_state    = drv;
1965
1966    normal->open        = uart_open;
1967    normal->close       = uart_close;
```

```
1968    normal->write       = uart_write;
1969    normal->put_char    = uart_put_char;
1970    normal->flush_chars = uart_flush_chars;
1971    normal->write_room  = uart_write_room;
1972    normal->chars_in_buffer = uart_chars_in_buffer;
1973    normal->flush_buffer    = uart_flush_buffer;
1974    normal->ioctl       = uart_ioctl;
1975    normal->throttle    = uart_throttle;
1976    normal->unthrottle  = uart_unthrottle;
1977    normal->send_xchar  = uart_send_xchar;
1978    normal->set_termios = uart_set_termios;
1979    normal->stop        = uart_stop;
1980    normal->start       = uart_start;
1981    normal->hangup      = uart_hangup;
1982    normal->break_ctl   = uart_break_ctl;
1983    normal->wait_until_sent = uart_wait_until_sent;
1984   #ifdef CONFIG_PROC_FS
1985    normal->read_proc   = uart_read_proc;
1986   #endif
1987
1988    /*
1989     * The callout device is just like the normal device except for
1990     * the major number and the subtype code.
1991     */
1992    *callout        = *normal;
1993    callout->name       = drv->callout_name;
1994    callout->major      = drv->callout_major;
1995    callout->subtype    = SERIAL_TYPE_CALLOUT;
1996    callout->read_proc  = NULL;
1997    callout->proc_entry = NULL;
1998
1999    for (i = 0; i < drv->nr; i++) {
2000        struct uart_state *state = drv->state + i;
2001
2002        state->callout_termios  = callout->init_termios;
2003        state->normal_termios   = normal->init_termios;
2004        state->port     = drv->port + i;
2005        state->port->line   = i;
2006
2007        uart_setup_port(drv, state);
2008    }
2009
2010    retval = tty_register_driver(normal);
2011    if (retval)
2012        goto out;
2013
2014    retval = tty_register_driver(callout);
```

```
2015    if (retval)
2016        tty_unregister_driver(normal);
2017
2018    out:
2019    if (retval && drv->state)
2020        kfree(drv->state);
2021    return retval;
2022    }
2023
2024    void uart_unregister_driver(struct uart_driver *drv)
2025    {
2026    int i;
2027
2028    for (i = 0; i < drv->nr; i++) {
2029        struct uart_state *state = drv->state + i;
2030
2031        if (state->info && state->info->tty)
2032            tty_hangup(state->info->tty);
2033
2034        pm_unregister(state->pm);
2035
2036        if (state->port->type != PORT_UNKNOWN)
2037            state->port->ops->release_port(state->port);
2038        if (state->info) {
2039            tasklet_kill(&state->info->tlet);
2040            kfree(state->info);
2041        }
2042    }
2043
2044    tty_unregister_driver(drv->normal_driver);
2045    tty_unregister_driver(drv->callout_driver);
2046
2047    kfree(drv->state);
2048    }
2049
2050    static int uart_match_port(struct uart_port *port1, struct uart_port *port2)
2051    {
2052    if (port1->iotype != port2->iotype)
2053        return 0;
2054    switch (port1->iotype) {
2055    case SERIAL_IO_PORT:    return (port1->iobase == port2->iobase);
2056    case SERIAL_IO_MEM: return (port1->membase == port2->membase);
2057    }
2058    return 0;
2059    }
2060
2061    /**
```

```
2062    *   uart_register_port: register a port with the generic uart driver
2063    *   @reg: pointer to the uart low level driver structure for this port
2064    *   @port: uart port structure describing the port
2065    *
2066    *   Register a UART with the specified low level driver.  Detect the
2067    *   type of the port if ASYNC_BOOT_AUTOCONF is set, and detect the IRQ
2068    *   if ASYNC_AUTO_IRQ is set.
2069    *
2070    *   Returns negative error, or positive line number.
2071    */
2072   int uart_register_port(struct uart_driver *drv, struct uart_port *port)
2073   {
2074    struct uart_state *state = NULL;
2075    int i, flags = UART_CONFIG_TYPE;
2076
2077    /*
2078     * First, find a port entry which matches.  Note: if we do
2079     * find a matching entry, and it has a non-zero use count,
2080     * then we can't register the port.
2081     */
2082    down(&port_sem);
2083    for (i = 0; i < drv->nr; i++) {
2084        if (uart_match_port(drv->state[i].port, port)) {
2085            down(&drv->state[i].count_sem);
2086            state = &drv->state[i];
2087            break;
2088        }
2089    }
2090
2091    /*
2092     * If we didn't find a matching entry, look for the first
2093     * free entry.  We look for one which hasn't been previously
2094     * used (indicated by zero iobase).
2095     */
2096    if (!state) {
2097        for (i = 0; i < drv->nr; i++) {
2098            if (drv->state[i].port->type == PORT_UNKNOWN &&
2099                drv->state[i].port->iobase == 0) {
2100                down(&drv->state[i].count_sem);
2101                if (drv->state[i].count == 0) {
2102                    state = &drv->state[i];
2103                    break;
2104                }
2105            }
2106        }
2107    }
2108
```

```
2109    /*
2110     * Ok, that also failed.  Find the first unused entry, which
2111     * may be previously in use.
2112     */
2113    if (!state) {
2114        for (i = 0; i < drv->nr; i++) {
2115            if (drv->state[i].port->type == PORT_UNKNOWN) {
2116                down(&drv->state[i].count_sem);
2117                if (drv->state[i].count == 0) {
2118                    state = &drv->state[i];
2119                    break;
2120                }
2121            }
2122        }
2123    }
2124
2125    up(&port_sem);
2126
2127    if (!state)
2128        return -ENOSPC;
2129
2130    /*
2131     * If we find a port that matches this one, and it appears to
2132     * be in-use (even if it doesn't have a type) we shouldn't alter
2133     * it underneath itself - the port may be open and trying to do
2134     * useful work.
2135     */
2136    if (state->count != 0 ||
2137        (state->info && state->info->blocked_open != 0)) {
2138        up(&state->count_sem);
2139        return -EBUSY;
2140    }
2141
2142    /*
2143     * We're holding the lock for this port.  Copy the relevant data
2144     * into the port structure.
2145     */
2146    state->port->iobase   = port->iobase;
2147    state->port->membase  = port->membase;
2148    state->port->irq      = port->irq;
2149    state->port->uartclk  = port->uartclk;
2150    state->port->fifosize = port->fifosize;
2151    state->port->regshift = port->regshift;
2152    state->port->iotype   = port->iotype;
2153    state->port->flags    = port->flags;
2154
2155   #if 0 //def CONFIG_PM
```

```
2156    /* we have already registered the power management handlers */
2157    state->pm = pm_register(PM_SYS_DEV, PM_SYS_COM, uart_pm);
2158    if (state->pm) {
2159        state->pm->data = state;
2160
2161        /*
2162         * Power down all ports by default, except
2163         * the console if we have one.
2164         */
2165        if (!drv->cons || state->port->line != drv->cons->index)
2166            pm_send(state->pm, PM_SUSPEND, (void *)3);
2167    }
2168   #endif
2169
2170    if (state->port->flags & ASYNC_AUTO_IRQ)
2171        flags |= UART_CONFIG_IRQ;
2172    if (state->port->flags & ASYNC_BOOT_AUTOCONF)
2173        state->port->ops->config_port(state->port, flags);
2174
2175    tty_register_devfs(drv->normal_driver, 0, drv->minor +
2176                       state->port->line);
2177    tty_register_devfs(drv->callout_driver, 0, drv->minor +
2178                       state->port->line);
2179
2180    uart_report_port(drv, state->port);
2181
2182    up(&state->count_sem);
2183    return i;
2184   }
2185
2186   /*
2187    * Unregister the specified port index on the specified driver.
2188    */
2189   void uart_unregister_port(struct uart_driver *drv, int line)
2190   {
2191    struct uart_state *state;
2192
2193    if (line < 0 || line >= drv->nr) {
2194        printk(KERN_ERR "Attempt to unregister %s%d\n",
2195            drv->normal_name, line);
2196        return;
2197    }
2198
2199    state = drv->state + line;
2200
2201    down(&state->count_sem);
2202    /*
```

```
2203     * The port has already gone.  We have to hang up the line
2204     * to kill all usage of this port.
2205     */
2206    if (state->info && state->info->tty)
2207        tty_hangup(state->info->tty);
2208
2209    /*
2210     * Free the ports resources, if any.
2211     */
2212    state->port->ops->release_port(state->port);
2213
2214    /*
2215     * Indicate that there isn't a port here anymore.
2216     */
2217    state->port->type = PORT_UNKNOWN;
2218
2219   #if 0 // not yet
2220    /*
2221     * No point in doing power management for hardware that
2222     * isn't present.
2223     */
2224    pm_unregister(state->pm);
2225   #endif
2226
2227    /*
2228     * Remove the devices from devfs
2229     */
2230    tty_unregister_devfs(drv->normal_driver, drv->minor + line);
2231    tty_unregister_devfs(drv->callout_driver, drv->minor + line);
2232    up(&state->count_sem);
2233   }
2234
2235   EXPORT_SYMBOL(uart_event);
2236   EXPORT_SYMBOL(uart_register_driver);
2237   EXPORT_SYMBOL(uart_unregister_driver);
2238   EXPORT_SYMBOL(uart_register_port);
2239   EXPORT_SYMBOL(uart_unregister_port);
2240
2241   static int __init uart_init(void)
2242   {
2243    return 0;
2244   }
2245
2246   static void __exit uart_exit(void)
2247   {
2248    free_page((unsigned long)tmp_buf);
2249    tmp_buf = NULL;
```

```
2250    }
2251
2252    module_init(uart_init);
2253    module_exit(uart_exit);
2254
2255    MODULE_DESCRIPTION("Serial driver core");
2256    MODULE_LICENSE("GPL");
```

uClinux-dist/vendor/Samsung/44B0X/config.arch

```
1   .EXPORT_ALL_VARIABLES:
2   ###############################################################################
3   #
4   # Vendor specific settings
5   #
6
7   CONSOLE_BAUD_RATE = 19200
8
9   ###############################################################################
10
11   CPUFLAGS :=
12   VENDOR_CFLAGS :=
13   DISABLE_XIP := 1
14   DISABLE_MOVE_RODATA := 1       # until we get a toolchain release
15   # LOPT := -O2                  # library is alway O2
16   # UOPT := -Os                  # user apps are always Os
17
18   ###############################################################################
19
20   include $(ROOTDIR)/vendors/config/armnommu/config.arch
21
22   ###############################################################################
```

uClinux-dist/vendor/Samsung/44B0X/Makefile

```
#
#   Makefile -- Build instructions for Samsung/4510B
#

ROMFSIMG = $(IMAGEDIR)/romfs.img
ROMIMAGE = $(IMAGEDIR)/image.rom
RAMIMAGE = $(IMAGEDIR)/image.ram


 ROMFS_DIRS = bin dev etc home lib mnt proc usr var

 DEVICES = \
    tty,c,5,0   console,c,5,1   cua0,c,5,64 cua1,c,5,65 \
    \
    mem,c,1,1   kmem,c,1,2  null,c,1,3 \
    zero,c,1,5  random,c,1,8    urandom,c,1,9  \
    \
    ram0,b,1,0  ram1,b,1,1 \
    \
    ptyp0,c,2,0 ptyp1,c,2,1 ptyp2,c,2,2 ptyp3,c,2,3 \
    ptyp4,c,2,4 ptyp5,c,2,5 ptyp6,c,2,6 ptyp7,c,2,7 \
    ptyp8,c,2,8 ptyp9,c,2,9 ptypa,c,2,10    ptypb,c,2,11 \
    ptypc,c,2,12    ptypd,c,2,13    ptype,c,2,14    ptypf,c,2,15 \
    \
    rom0,b,31,0 rom1,b,31,1 rom2,b,31,2 rom3,b,31,3 \
    rom4,b,31,4 rom5,b,31,5 rom6,b,31,6 rom7,b,31,7 \
    rom8,b,31,8 rom9,b,31,9 \
    \
    tty0,c,4,0  tty1,c,4,1  tty2,c,4,2  tty3,c,4,3 \
    ttyS0,c,4,64    ttyS1,c,4,65 \
    \
    ttyp0,c,3,0 ttyp1,c,3,1 ttyp2,c,3,2 ttyp3,c,3,3 \
    ttyp4,c,3,4 ttyp5,c,3,5 ttyp6,c,3,6 ttyp7,c,3,7 \
    ttyp8,c,3,8 ttyp9,c,3,9 ttypa,c,3,10    ttypb,c,3,11 \
    ttypc,c,3,12    ttypd,c,3,13    ttype,c,3,14    ttypf,c,3,15


clean:

romfs:
    [ -d $(ROMFSDIR)/$$i ] || mkdir -p $(ROMFSDIR)
    for i in $(ROMFS_DIRS); do \
        [ -d $(ROMFSDIR)/$$i ] || mkdir -p $(ROMFSDIR)/$$i; \
    done
    for i in $(DEVICES); do \
```

```
46          touch $(ROMFSDIR)/dev/@$$i; \
47      done
48      $(ROMFSINST) -s /var/tmp /tmp
49      $(ROMFSINST) -s /bin /sbin
50      $(ROMFSINST) /etc/rc
51      $(ROMFSINST) /etc/inittab
52      $(ROMFSINST) ../../Generic/romfs/etc/services /etc/services
53      case "$(LINUXDIR)" in \
54      *2.4.*) ;; \
55      *) echo "ttyS0:linux:/bin/sh" >> $(ROMFSDIR)/etc/inittab ;; \
56      esac
57      $(ROMFSINST) /etc/motd
58  #   $(ROMFSINST) /etc/boa.conf
59  #   $(ROMFSINST) index.html /home/index.html
60      mkdir -p $(ROMFSDIR)/home/httpd
61      $(ROMFSINST) index.html /home/httpd/index.html
62      $(ROMFSINST) eda.gif /home/httpd/eda.gif
63      $(ROMFSINST) boa_logo1.png /home/httpd/boa_logo1.png
64      $(ROMFSINST) ucllogo.gif /home/httpd/ucllogo.gif
65      $(ROMFSINST) /etc/passwd
66      echo "$(VERSIONSTR) -- " `date` > $(ROMFSDIR)/etc/version
67
68  image:
69      [ -d $(IMAGEDIR) ] || mkdir -p $(IMAGEDIR)
70      genromfs -v -V "ROMdisk" -f $(ROMFSIMG) -d $(ROMFSDIR)
71      $(CROSS_COMPILE)ld -r -o $(ROOTDIR)/$(LINUXDIR)/romfs.o \
72              -b binary $(ROMFSIMG)
73      $(CROSS_COMPILE)objcopy -O binary -R .note -R .comment \
74              -S $(ROOTDIR)/$(LINUXDIR)/linux $(RAMIMAGE)
75      cp $(ROOTDIR)/$(LINUXDIR)/arch/armnommu/boot/zImage \
76              $(ROMIMAGE)
```

uClinux-dist/vendor/Samsung/44B0X/motd

```
Welcome to
          ____ _  _
         /  __| ||_|
        _|  |  | | _ ____  _   _  _  _
       | | | | |  | || |  _ \| | | |\ \/ /
       | |_| | |__| || | | | | |_| |/    \
       |  ___\____|_||_|_| |_|\____|\_/\_/
       | |
       |_|

 For further information check:
 http://www.uclinux.org/

 This port can be run under 51EDA's ARM Development Board
 http://www.51eda.com/

```

uClinux-dist/vendor/Samsung/44B0X/rc

```
hostname Samsung
/bin/expand /etc/ramfs.img /dev/ram0
mount -t proc proc /proc
mount -t ext2 /dev/ram0 /var
mkdir /var/config
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
 mkdir /var/empty
 cat /etc/motd
```

(5) 补丁的制作

补丁文件的存在，给了我们一条方便的途径给内核源代码进行修改。uClinux 官方发行版经过自己的修改之后中，就可以修改补丁文件了。假定 uClinux-51eda 是我们已经修改过的源代码所在的目录，uClinux-dist 存放未经修改的源代码。则执行下面的指令将可生成补丁文件—uclinux-51eda.patch：

diff -urN uClinux-51eda uClinux-dist > uclinux-51eda.patch

说明一下 diff 的参数项：

-n 输出 RC-格式的 diffs；

-r 当比较目录时，递归比较任何找到的子项录。

-N --new-file 在目录比较中，如果那个文件只在其中一个目录中找到，那么它被视为在另一个目录是一个空文件。

一般在进行制作补丁前，需要把我们之前编译生成的文件删除。这可以通过执行

make clean

make mrproper

来实现。当有了一个恰当版本的 uClinux 官方发行版，只要打上我们制作的补丁文件，则可以完成移植。给源码打补丁的过程如下：

将 uclinux-51eda.pach 文件拷贝到官方发行版的 uClinux-dist 中

patch -p1 < uclinux-51eda.patch

这样就可以完成了。这个补丁文件同样在配套光盘\uClinux_porting 提供。用户可以尝试在我们提供的 uClinux 官方发行版的基础上打此补丁。

附录

附录 A　　vi 使用方法简介

在 Linux 的 Shell 环境下进行程序的编写和修改，推荐使用的工具是 vi 。它是 UNIX & Linux 系统自带的文本编辑器，具有强大的指令编辑功能，能方便快捷的完成文本的编辑和修改工作。由于 vi 编辑器因版本等原因会有所不同，所以以下介绍中如有出入，请参阅相关帮助并实践检验。

Vi 编辑器的指令基本在命令模式中输入。命令模式下有如下指令供使用:（任何时候按 Escape 键可回到命令模式）

h 将光标左移一格

l 将光标右移一格

j 将光标下移一格

k 将光标上移一格

w 将光标移到下一个小字的前面

W 将光标移到下一个大字的前面

b 将光标移到前一个小字的前面

B 将光标移到前一个大字的前面

e 将光标移到下一个小字的后面

E 将光标移到下一个大字的后面

fc 将光标移到同一行的下一个字符 c 处

Fc 将光标移到同一行的前一个字符 c 处

tc 将光标移到同一行的下一个字符 c 的前一格

Tc 将光标移到同一行的前一个字符 c 的后一格

number| 将光标移到第 number 列上

+或者 Enter 将光标移到下一行第一个非空白字符处。

- 将光标移到上一行第一个非空白字符处。

0 将光标移到当前行的第一个字符处

$ 将光标移到当前行的最后一个字符处

H 将光标移到屏幕最顶端一行

L 将光标移到屏幕最底端一行

M 将光标移到屏幕的中间

z- 把当前行作为屏幕的最后一行，刷新屏幕

z. 把当前行作为屏幕的中间一行，刷新屏幕

Ctrl+l 重新显示屏幕当前内容

Ctrl+f 向后滚一页

Ctrl+d 向后滚半页

Ctrl+b 向前滚一页

Ctrl+u 向前滚半页

Ctrl+e 屏幕向下滚一行

Ctrl+y 屏幕向上滚一行

/pattern 向后寻找指定字符串 pattern

?pattern 向前寻找指定字符串 pattern

n 在上次指定的方向上，再次查找

N 在上次指定的方向的反方向上，再次查找

% 移到匹配的“()”或者“{}”上

a 光标移到所在处之后，进入文本输入状态

A 光标移到行尾，进入文本输入状态

i 在光标所在处，进入文本输入状态

I 光标移到行首第一个非空白的字符处，进入文本输入状态

o 在光标所在行后插入一空行，进入文本输入状态

O 在光标所在行前插入一空行，进入文本输入状态

cc 或者 S 将当前一行清空，进入文本输入状态，覆盖方式

C 改变本行光标以后的文字，进入文本输入状态，覆盖方式

cw 改变光标所在位置的单词，进入文本输入状态，覆盖方式

dd 删除当前行，后续行自动上移

D 删除光标所在行光标以后的文字

Dw 删除光标所在单词

J 把下一行内容加到本行行尾

rc 把光标处字符替换成 c

R 覆盖本行内容，本行编辑模式变为改写模式

u 恢复上一次的修改

x 删除光标所在处字符

X 删除光标左侧字符

~ 改变光标所在处字符的大小写

. 重复上一次操作

<< 当前行左移一个 Tab

>> 当前行右移一个 Tab

yy 或者 Y 把当前行放入缓冲区

yw 将光标所在单词放入缓冲区

p 将缓冲区内容放入光标所在行的下面

P 将缓冲区内容放入光标所在行的上面

:w 回写修改后的文件

:w filename 当 filename 不存在，写成 filename，否则报错

:w! filename filename 存在也直接写成 filename

:n 开始编辑 vi 激活的文件列表中的下一个文件

:n filename 开始编辑指定的文件 filename

:e filename 使用 filename 激活 vi，在 vi 中装入另外一个文件 filename

:e! 重新装入当前文件，丢弃一切未保存的改动

:r filename 读取 filename 的内容，加在光标处

:r! command 执行 command 文件，将其输出加在光标处

Ctrl+g 取得正在编辑的文件的有关信息

:sh 起动 sh，从 sh 中返回可以用 exit 或者 Ctrl-d

:! command 执行 command 指令

:!! 重新执行上次的! command 指令

:q 退出 vi，如果用户未将修改保存，不能退出

:q! 退出 vi，不管是否有未保存的修改

:wq 或者:x 退出 vi 并保存修改

附录 B　uClinux 中断号

在 uClinux 中，44B0X 处理器各中断所对应的中断号在 uClinux-dist\linux-2.4.x\include\asm-armnommu\arch-s3c44b0\irqs.h 文件中定义如下：

```
#define INT_GLOBAL       26
#define INT_EXTINT0      25
#define INT_EXTINT1      24
#define INT_EXTINT2      23
#define INT_EXTINT3      22
#define    INT_EXTINT4_7 21
#define    INT_TICK      20
#define    INT_ZDMA0     19
#define    INT_ZDMA1     18
#define    INT_BDMA0     17
#define    INT_BDMA1     16
#define    INT_WDT       15
#define    INT_UERR0_1   14
#define    INT_TIMER0    13
#define    INT_TIMER1    12
#define    INT_TIMER2    11
#define    INT_TIMER3    10
#define    INT_TIMER4    9
```

附录 C　XMODEM 协议

XMODEM 协议是一种使用拨号调制解调器的个人计算机通信中广泛使用的异步文件运输协议。这种协议以 128 字节块的形式传输数据，并且每个块都使用一个校验和过程来进行错误检测。使用 PC 机传送文件的早期协议。在数据传送过程中使用 1

字节的控制序列，以便通讯双方对传送过程进行监视。如果接收方关于一个块的校验和与它在发送方的校验和相同时，接收方就向发送方发送一个认可字节。然而，这种对每个块都进行认可的策略将导致低性能，特别是具有很长传播延迟的卫星连接的情况时，问题更加严重。

使用循环冗余校验的与 XMODEM 相应的一种协议称为 XMODEM－CRC 。还有一种是 XMODEM－IK ，它以 1024 字节一块来传输数据。ZMODEM 是最有效的一个 XMODEM 版本，它不需要对每个块都进行认可。事实上，它只是简单地要求对损坏的块进行重发。ZMODEM 对按块收费的分组交换网络是非常有用的。不需要认可回送分组在很大程度上减少了通信量。

YMODEM 也是一种 XMODEM 的实现。它包括 XMODEM－IK 的所有特征，另外在一次单一会话期间为发送一组文件，增加了批处理文件传输模式。

附录 D　Nand Flash 和 Nor Flash 详解

NOR 和 NAND 是现在市场上两种主要的非易失闪存技术。Intel 于 1988 年首先开发出 NOR flash 技术，彻底改变了原先由 EPROM 和 EEPROM 一统天下的局面。紧接着，1989 年，东芝公司发表了 NAND flash 结构，强调降低每比特的成本，更高的性能，并且象磁盘一样可以通过接口轻松升级。但是经过了十多年之后，仍然有相当多的硬件工程师分不清 NOR 和 NAND 闪存。

“flash 存储器”经常可以与“NOR 存储器互换使用。许多业内人士也搞不清楚 NAND 闪存技术相对于 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 的特点是芯片内执行（XIP, eXecute In Place），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。

性能比较

flash 闪存是非易失存储器，可以称为块的存储器单元块进行擦写和再编程。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 则要求在进行擦除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 64－128KB 的块进行的，执行一个写入／擦除操作的时间为 55ms，与此相反，擦除 NAND 器件是以 8－32KB 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 NADN 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），更多的擦除操作必须在基于 NOR 的单元中进行。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 55ms 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。

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。

NAND 器件使用复杂的 FO 口来串行地存取数据，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 读和写操作采用 512 字节的块，这一点有点像硬盘管理此类操作，很自然地，基于 NAND 的存储器就可以取代硬盘或其他块设备。

容量与成本

NAND flash 的单元尺寸几乎是 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。

NOR flash 占据了容量为 1－16MB 闪存市场的大部分，而 NAND flash 只是用在 8－128MB 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 secure Digital、Pc cards 和 MMC 存储卡市场上所占份额最大。

可靠性与耐用性

采用 flahs 介质时一个需要重点考虑的问题是可靠性。对于需要扩展 MTBF 的系统来说，Flash 是非常合适的存储方案。可以从寿命（耐用性）、位交换和坏块处理三个方面来比较 NOR 和 NAND 的可靠性。

寿命（耐用性）

在 NAND 闪存中每个块的最大擦写次数是一百万次，而 NOR 的擦写次数是十万次。NAND 存储器除了具有 10 比 1 的块擦除周期优势，典型的 NAN 块尺寸要比 NOR 器件小 8 倍，每个 NAND 存储器块在给定的时间内的删除次数要少一些。

位交换

所有 flash 器件都受位交换现象的困扰。在某些情况下（很少见，NAND 发生的次数要比 NOR 多），一个比特位会发生反转或被报告反转了。

一位的变化可能不很明显，但是如果发生在一个关键文件上，这个小小的故障

可能导致系统停机。如果只是报告有问题，多读几次就可能解决了。

当然，如果这个位真的改变了，就必须采用错误探测 / 错误更正（EDC/ECC ）算法。位反转的问题更多见于 NAND 闪存，NAND 的供应商建议使用 NAND 闪存的时候，同时使用 EDC/ECC 算法。

这个问题对于用 NAND 存储多媒体信息时倒不是致命的。当然，如果用本地存储设备来存储操作系统、配置文件或其他敏感信息时，必须使用 EDC/ECC 系统以确保可靠性。

坏块处理

NAND 器件中的坏块是随机分布的。以前也曾有过消除坏块的努力，但发现成品率太低，代价太高，根本不划算。

NAND 器件需要对介质进行初始化扫描以发现坏块，并将坏块标记为不可用。在已制成的器件中，如果通过可靠的方法不能进行这项处理，将导致高故障率。

易于使用

可以非常直接地使用基于 NOR 的闪存，可以像其他存储器那样连接，并可以在上面直接运行代码。

由于需要 FO 接口，NAND 要复杂得多。各种 NAND 器件的存取方法因厂家而异。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，因为设计师绝不能向坏块写入，这就意味着在 NAND 器件上自始至终都必须进行虚拟映射。

软件支持

当讨论软件支持的时候，应该区别基本的读 / 写 / 擦操作和高一级的用于磁盘仿真和闪存管理算法的软件，包括性能优化。

在 NOR 器件上运行代码不需要任何的软件支持，在 NAND 器件上进行同样操作时，通常需要驱动程序，也就是内存技术驱动程序（MTD），NAND 和 NOR 器件在进行写入和擦除操作时都需要 MTD。

使用 NOR 器件时所需要的 MTD 要相对少一些，许多厂商都提供用于 NOR 器件的更高级软件，这其中包括 M-System 的 TrueFFS 驱动，该驱动被 Wind River system、Microsoft、QNX、Sylllbia 和 Inte 等厂商所采用。

驱动还用于对 DiskonChip 产品进行仿真和 NAND 闪存的管理，包括纠错、坏块处理和损耗平衡。

附录 E　YAFFS 文件系统

YAFFS 是 Yet Another Flash File System 的简写，YAFFS 文件系统跟 JFFS2 文件系统一样，也是一种基于 Flash 的日志文件系统，能在掉电时自动提供可靠的数据记录、恢复，防止自身文件系统的崩溃，并能对坏块进行管理。它用独立的日志文件跟踪磁盘内容的变化。举例来说：当驱动程序需要写存储器中某一块时，它修改的是存放于文件日志中的一块镜像；只有当日志中的镜像复制到日志文件系统中后，数据才真实地写到了那一块上。这种技术允许撤消对文件系统做的改变(只要把日志中的数据覆盖)，或者在发生崩溃时恢复数据(只要把镜像拷贝到文件系统中)。日志文件系统提供了更好的性能表现(数据是并行地写入进日志,然后再逐步写入文件系统)，和更好的崩溃恢复性能。它在设计时充分考虑了 FLASH 的读写特性，专门提供了 NAND Flash 芯片的优化技术.当然它对 NOR Flash 和 RAM 也提供了良好的支持。另外它也确保在读取文件时，如果系统突然掉电，其文件的可靠性不受到影响。当然，YAFFS 对 Flash 的读写采用 ECC 校验，增强数据访问的安全性。

附录 F　uClinux 的多进程分析

uClinux 没有 MMU 管理存储器，在实现多个进程时(fork 调用生成子进程)需要实现数据保护。uClinux 的 fork 等于 vfork。实际上 uClinux 的多进程管理是通过 vfork 来实现。这意味着 uclinux 系统 fork 调用完成后，要么子进程代替父进程执行(此时父进程已经 sleep)直到子进程调用 exit 退出;要么调用 exec 执行一个新的进程，这个时候将产生可执行文件的加载，即使这个进程只是父进程的拷贝，这个过程也不能避免。当子进程执行 exit 或 exec 后,子进程使用 wakeup 把父进程唤醒，使父进程继续往下执行。实际上，应该说 MMU 使得多任务的实现更为简单和高效。比如 DOS 没有启动 MMU，但一样可以部分支持多任务，比如：后台打印。其实，多任务的主要问题是如何进行进程上下文的安全切换(包括重定位和保护)。如果 fork 实现为完全拷贝一份新的内存副本，没有 MMU 一样可以支持多任务。实现上，没有 MMU 的主要问题是 fork 如何优化实现的问题。fork 系统调用必须给子进程提供一个与父进程地址空间逻辑上明显不同的副本。大多数情况下，当子进程在 fork 后调用 exec 或 exit，它会遗弃这个地址空间。因此，在以前的 UNIX 系统中，为进程建立一个实际的地址空间副本是非常浪费的。这个问题通过两种方法来解决。第一种是 copy-on-write(写时复制)方法，它首先由 System V 采用，现在为包括 Linux 在内的大多数 UNIX 系统采用。在这种方法中，父进程的数据和堆栈页暂时设置为只读，并被标记为 copy-on-write。子进程有一份自己的地址转换表(比如页表)，与父进程共享内存页。而无论是父进程还是子进程试图修改页时，都会产生页面失效异常(由于页表标识为只读)并且会调用内核的失效处理程序(Linux：do_wp_page)。该处理程序识别这是一个 copy-on-write 页，为其建立一个新的可写的页副本。由此只有那些被修改的页才需复制，而不是全部页面。若子进程调用 exec 或 exit，页面回到原来的保护方式，copy-on-write 标识被清除。

BSD 提供了另一种解决方案即新的 vfork 系统调用。若用户进程希望在调用 fork 后随即调用 exec，它可以调用 vfork 而不是 fork。vfork 不进行复制。相反，父进程将自己的地址空间租借给子进程，并将自己阻塞，直到子进程将地址空间归还给它。因此子进程会一直使用父进程的地址空间直到它调用 exec 或 exit。于是内核

将地址空间返回给父进程并唤醒它。vfork 非常快，甚至连地址映射表(如页表)也无需复制。地址空间传给子进程只是简单地拷贝地址映射寄存器。然而，由于它允许一个进程使用并修改另一个进程的地址空间，这是一个非常危险的调用。某些程序，如 csh 就是利用这个特性。

附录 G　test 驱动程序源代码

```
#ifndef __KERNEL__
#define __KERNEL__
#endif

#ifndef MODULE
#define MODULE
#endif

#include <linux/module.h>
#include <linux/kernel.h>      /* For printk() */
#include <linux/config.h>
#include <linux/fs.h>          /* For struct file_operations */
#include <linux/types.h>
#include <linux/sched.h>
#include <linux/init.h>

#define TEST_MAJOR    253

static ssize_t test_read(struct file *filp, char* buf, size_t count, loff_t *f_pos);
static ssize_t test_write(struct file *filp, const char* buf, size_t count, loff_t *f_pos);
static loff_t test_llseek(struct file *filp, loff_t off, int whence);
static int test_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg);
static int test_open(struct inode *inode, struct file *filp);
static int test_release(struct inode *inode, struct file *filp);

/* 以下为驱动程序口的核心数据结构，它告诉系统内核，该设备的几个操作所对应的函数入口点
    在该结构体的定义中采用了标记化的结构初始化语法 */
struct file_operations test_fops = {
      llseek:     test_llseek,
      read:test_read,
      write:      test_write,
      ioctl:      test_ioctl,
      open:       test_open,
      release: test_release,
};


/*    驱动程序 open 操作对应的操作函数 */
static int test_open(struct inode *inode, struct file *filp)
{
      printk("Device TEST open!\n\r");
      return 0;
}

/*    驱动程序 llseek 操作对应的操作函数 */
static loff_t test_llseek(struct file *filp, loff_t off, int whence)
{
      printk("Device TEST seek!\n\r");
```

```
    return 1;
}

/*  驱动程序 write 操作对应的操作函数 */
static ssize_t test_write(struct file *flip, const char *buf, size_t count, loff_t *f_pos)
{
    printk("Device TEST write!\n\r");
    return count;
}

/*  驱动程序 ioctl 操作对应的操作函数  */
static int test_ioctl(struct inode *inode, struct file *filp, unsigned int cmd, unsigned long arg)
{
    printk("Device TEST ioctl!\n\r");
    return 1;
}

/*  驱动程序 release 操作对应的操作函数      */
static int test_release(struct inode *inode, struct file *flip)
{
    printk("Device TEST release!\n\r");
    return 0;
}

/*  驱动程序 read 操作对应的操作函数  */
static ssize_t test_read(struct file *filp, char *buf, size_t count, loff_t *f_pos)
{
    printk("Device TEST read!\n\r");
    return count;
}

/*  驱动程序真正的初始化过程，它由本程序最后的 module_init 指明
    由于它使用了__init 关键字，而且该驱动程序是以静态方式编译到内核中的，
    因此当驱动程序初始化完成后，它所占据的内存空间会被释放   */
static int __init test_init_module(void)
{
    int result;

    printk("Device TEST init_module\n\r");
    result = register_chrdev(TEST_MAJOR, "TEST", &test_fops);

    if (result < 0) {
        printk("TEST can't get major number\n\r");
        return result;
    }
    return 0;

}

/*  驱动程序真正的退出过程，它由本程序最后的 module_cleanup 指明
    由于它使用了__exit 关键字，并且，它也是静态编译至内核中的，
    因此，下面这段代码都不会被执行，因为以静态方式编译到内核的驱动程序
```

```
    是无法卸载的      */
static void __exit test_cleanup_module(void)
{
    unregister_chrdev(TEST_MAJOR, "TEST");
    printk("Device TEST remove\n\r");
}

module_init(test_init_module);              //驱动程序入口函数
module_exit(test_cleanup_module);           //驱动程序出口函数

EXPORT_NO_SYMBOLS;                          //为了不把该驱动程序中定义的符号，变量名等
                                //引入内核空间，我们需要在驱动程序中加入这个宏
MODULE_LICENSE("GPL");                       //驱动程序的版权说明
MODULE_AUTHOR("51EDA");                      //指明该驱动程序模块的作者
```

附录 H　Make 和 Makefile 详解

（本文摘自陈皓的 BLOG:http://blog.csdn.net/haoel/）

编者按：在 Makefile 的中文文档中，鲜有出其右者，特收录在本书的附录中

概述

什么是 Makefile？或许很多 Windows 的程序员都不知道这个东西，因为那些 Windows 的 IDE 都为你做了这个工作，但我觉得要作一个好的和专业的程序员，Makefile 还是要懂。这就好像现在有这么多的 HTML 的编辑器，但如果你想成为一个专业人士，你还是要了解 HTML 的标识的含义。特别在 Unix 和 Linux 环境下的软件编译，你就不能不自己写 Makefile 了，会不会写 Makefile，从一个侧面说明了一个人是否具备完成大型工程的能力。

因为，Makefile 关系到了整个工程的编译规则。一个工程中的源文件不计其数，其按类型、功能、模块分别放在若干个目录中，Makefile 定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，因为 Makefile 就像一个 Shell 脚本一样，其中也可以执行操作系统的命令。

Makefile 带来的好处就是——“自动化编译”，一旦写好，只需要一个 make 命令，整个工程完全自动编译，极大的提高了软件开发的效率。make 是一个命令工具，是一个解释 Makefile 中指令的命令工具，一般来说，大多数的 IDE 都有这个命令，比如：Delphi 的 make，Visual C++ 的 nmake，Linux 下 GNU 的 make。可见，Makefile 都成为了一种在工程方面的编译方法。

在这篇文章中，将以 C/C++的源码作为我们基础，所以必然涉及一些关于 C/C++的编译的知识，相关于这方面的内容，还请各位查看相关的编译器的文档。这里所默认的编译器是 UNIX 下的 GCC 和 CC。

关于程序的编译和链接

在此，我想多说关于程序编译的一些规范和方法，一般来说，无论是 C、C++、还是 pas，首先要把源文件编译成中间代码文件，在 Windows 下也就是 .obj 文件，UNIX 下是 .o 文件，即 Object File，这个动作叫做编译（compile）。然后再把大量的 Object File 合成执行文件，这个动作叫作链接（link）。

编译时，编译器需要的是语法的正确，函数与变量的声明的正确。对于后者，通常是你需要告诉编译器头文件的所在位置（头文件中应该只是声明，而定义应该放在 C/C++文件中），只要所有的语法正确，编译器就可以编译出中间目标文件。一般来说，每个源文件都应该对应于一个中间目标文件（O 文件或是 OBJ 文件）。

链接时，主要是链接函数和全局变量，所以，我们可以使用这些中间目标文件（o 文件或是 OBJ 文件）来链接我们的应用程序。链接器并不管函数所在的源文件，只管函数的中间目标文件（Object File），在大多数时候，由于源文件太多，编译生成的中间目标文件太多，而在链接时需要明显地指出中间目标文件名，这对于编译很不方便，所以，我们要给中间目标文件打个包，在 Windows 下这种包叫“库文件”（Library File），也就是 .lib 文件，在 UNIX 下，

是 Archive File，也就是 .a 文件。

总结一下，源文件首先会生成中间目标文件，再由中间目标文件生成执行文件。在编译时，编译器只检测程序语法，和函数、变量是否被声明。如果函数未被声明，编译器会给出一个警告，但可以生成 Object File。而在链接程序时，链接器会在所有的 Object File 中找寻函数的实现，如果找不到，那到就会报链接错误码（Linker Error），在 VC 下，这种错误一般是：Link 2001 错误，意思说是说，链接器未能找到函数的实现。你需要指定函数的 Object File.

Makefile 介绍

make 命令执行时，需要一个 Makefile 文件，以告诉 make 命令需要怎么样的去编译和链接程序。

首先，我们用一个示例来说明 Makefile 的书写规则。以便给大家一个感性认识。这个示例来源于 GNU 的 make 使用手册，在这个示例中，我们的工程有 8 个 C 文件，和 3 个头文件，我们要写一个 Makefile 来告诉 make 命令如何编译和链接这几个文件。我们的规则是：

1）如果这个工程没有编译过，那么我们的所有 C 文件都要编译并被链接。

2）如果这个工程的某几个 C 文件被修改，那么我们只编译被修改的 C 文件，并链接目标程序。

3）如果这个工程的头文件被改变了，那么我们需要编译引用了这几个头文件的 C 文件，并链接目标程序。

只要我们的 Makefile 写得够好，所有的这一切，我们只用一个 make 命令就可以完成，make 命令会自动智能地根据当前的文件修改的情况来确定哪些文件需要重编译，从而自己编译所需要的文件和链接目标程序。

一、Makefile 的规则

在讲述这个 Makefile 之前，还是让我们先来粗略地看一看 Makefile 的规则。

```
target ... : prerequisites ...
        command
        ...
        ...
```

target 也就是一个目标文件，可以是 Object File，也可以是执行文件。还可以是一个标签（Label），对于标签这种特性，在后续的“伪目标”章节中会有叙述。

prerequisites 就是要生成那个 target 所需要的文件或是目标。

command 也就是 make 需要执行的命令。（任意的 Shell 命令）

这是一个文件的依赖关系，也就是说，target 这一个或多个的目标文件依赖于 prerequisites 中的文件，其生成规则定义在 command 中。说白一点就是说，prerequisites 中如果有一个以上的文件比 target 文件要新的话，command 所定义的命令就会被执行。这就是 Makefile 的规则。也就是 Makefile 中最核心的内容。

二、一个示例

正如前面所说的，如果一个工程有 3 个头文件，和 8 个 C 文件，我们为了完成前面所述的

那三个规则，我们的 Makefile 应该是下面的这个样子的。

```
edit : main.o kbd.o command.o display.o \
       insert.o search.o files.o utils.o
        cc -o edit main.o kbd.o command.o display.o \
                   insert.o search.o files.o utils.o
main.o : main.c defs.h
        cc -c main.c
kbd.o : kbd.c defs.h command.h
        cc -c kbd.c
command.o : command.c defs.h command.h
        cc -c command.c
display.o : display.c defs.h buffer.h
        cc -c display.c
insert.o : insert.c defs.h buffer.h
        cc -c insert.c
search.o : search.c defs.h buffer.h
        cc -c search.c
files.o : files.c defs.h buffer.h command.h
        cc -c files.c
utils.o : utils.c defs.h
        cc -c utils.c
clean :
        rm edit main.o kbd.o command.o display.o \
           insert.o search.o files.o utils.o
```

反斜杠（\）是换行符的意思。这样比较便于 Makefile 的易读。我们可以把这个内容保存在文件为“Makefile”或“makefile”的文件中，然后在该目录下直接输入命令“make”就可以生成执行文件 edit。如果要删除执行文件和所有的中间目标文件，那么，只要简单地执行一下“make clean”就可以了。

在这个 Makefile 中，目标文件（target）包含：执行文件 edit 和中间目标文件（*.o），依赖文件（prerequisites）就是冒号后面的那些 .c 文件和 .h 文件。每一个 .o 文件都有一组依赖文件，而这些 .o 文件又是执行文件 edit 的依赖文件。依赖关系的实质上就是说明了目标文件是由哪些文件生成的，换言之，目标文件是哪些文件更新的。

在定义好依赖关系后，后续的那一行定义了如何生成目标文件的操作系统命令，一定要以一个 Tab 键作为开头。记住，make 并不管命令是怎么工作的，他只管执行所定义的命令。make 会比较 targets 文件和 prerequisites 文件的修改日期，如果 prerequisites 文件的日期要比 targets 文件的日期要新，或者 target 不存在的话，那么，make 就会执行后续定义的命令。

这里要说明一点的是，clean 不是一个文件，它只不过是一个动作名字，有点像 C 语言中的 lable 一样，其冒号后什么也没有，那么，make 就不会自动去找文件的依赖性，也就不会自动执行其后所定义的命令。要执行其后的命令，就要在 make 命令后明显得指出这个 lable 的名字。这样的方法非常有用，我们可以在一个 Makefile 中定义不用的编译或是和编译无关的命令，比如程序的打包，程序的备份，等等。

三、make 是如何工作的

在默认的方式下，也就是我们只输入 make 命令。那么，

1、make 会在当前目录下找名字叫“Makefile”或“makefile”的文件。

2、如果找到，它会找文件中的第一个目标文件（target），在上面的例子中，他会找到“edit”这个文件，并把这个文件作为最终的目标文件。

3、如果 edit 文件不存在，或是 edit 所依赖的后面的 .o 文件的文件修改时间要比 edit 这个文件新，那么，他就会执行后面所定义的命令来生成 edit 这个文件。

4、如果 edit 所依赖的.o 文件也存在，那么 make 会在当前文件中找目标为.o 文件的依赖性，如果找到则再根据那一个规则生成.o 文件。（这有点像一个堆栈的过程）

5、当然，你的 C 文件和 H 文件是存在的，于是 make 会生成 .o 文件，然后再用 .o 文件生命 make 的终极任务，也就是执行文件 edit 了。

这就是整个 make 的依赖性，make 会一层又一层地去找文件的依赖关系，直到最终编译出第一个目标文件。在找寻的过程中，如果出现错误，比如最后被依赖的文件找不到，那么 make 就会直接退出，并报错，而对于所定义的命令的错误，或是编译不成功，make 根本不理。make 只管文件的依赖性，即，如果在我找了依赖关系之后，冒号后面的文件还是不在，那么对不起，我就不工作啦。

通过上述分析，我们知道，像 clean 这种，没有被第一个目标文件直接或间接关联，那么它后面所定义的命令将不会被自动执行，不过，我们可以显示要 make 执行。即命令——“make clean”，以此来清除所有的目标文件，以便重编译。

于是在我们编程中，如果这个工程已被编译过了，当我们修改了其中一个源文件，比如 file.c，那么根据我们的依赖性，我们的目标 file.o 会被重编译（也就是在这个依性关系后面所定义的命令），于是 file.o 的文件也是最新的啦，于是 file.o 的文件修改时间要比 edit 要新，所以 edit 也会被重新链接了（详见 edit 目标文件后定义的命令）。

而如果我们改变了“command.h”，那么，kdb.o、command.o 和 files.o 都会被重编译，并且，edit 会被重链接。

四、Makefile 中使用变量

在上面的例子中，先让我们看看 edit 的规则：

```
edit : main.o kbd.o command.o display.o \
            insert.o search.o files.o utils.o
        cc -o edit main.o kbd.o command.o display.o \
                    insert.o search.o files.o utils.o
```

我们可以看到[.o]文件的字符串被重复了两次，如果我们的工程需要加入一个新的[.o]文件，那么我们需要在两个地方加（应该是三个地方，还有一个地方在 clean 中）。当然，我们的 Makefile 并不复杂，所以在两个地方加也不累，但如果 Makefile 变得复杂，那么我们就有可能会忘掉一个需要加入的地方，而导致编译失败。所以，为了 Makefile 的易维护，在 Makefile 中我们可以使用变量。Makefile 的变量也就是一个字符串，理解成 C 语言中的宏可能会更好。

比如，我们声明一个变量，叫 objects, OBJECTS, objs, OBJS, obj, 或是 OBJ，反正不管什么，只要能够表示 obj 文件就行了。我们在 Makefile 一开始就这样定义：

```
objects = main.o kbd.o command.o display.o \
          insert.o search.o files.o utils.o
```

于是，我们就可以很方便地在我们的 Makefile 中以“$(objects)”的方式来使用这个变量了，于是我们的改良版 Makefile 就变成下面这个样子：

```
objects = main.o kbd.o command.o display.o \
          insert.o search.o files.o utils.o

edit : $(objects)
        cc -o edit $(objects)
main.o : main.c defs.h
        cc -c main.c
kbd.o : kbd.c defs.h command.h
        cc -c kbd.c
command.o : command.c defs.h command.h
        cc -c command.c
display.o : display.c defs.h buffer.h
        cc -c display.c
insert.o : insert.c defs.h buffer.h
        cc -c insert.c
search.o : search.c defs.h buffer.h
        cc -c search.c
files.o : files.c defs.h buffer.h command.h
        cc -c files.c
utils.o : utils.c defs.h
        cc -c utils.c
clean :
        rm edit $(objects)
```

于是如果有新的 .o 文件加入，我们只需简单地修改一下 objects 变量就可以了。

五、让 make 自动推导

GNU 的 make 很强大，它可以自动推导文件以及文件依赖关系后面的命令，于是我们就没必要去在每一个[.o]文件后都写上类似的命令，因为，我们的 make 会自动识别，并自己推导命令。

只要 make 看到一个[.o]文件，它就会自动的把[.c]文件加在依赖关系中，如果 make 找到一个 whatever.o，那么 whatever.c，就会是 whatever.o 的依赖文件。并且 cc -c whatever.c 也会被推导出来，于是，我们的 Makefile 再也不用写得这么复杂。我们的是新的 Makefile 又出炉了。

```
objects = main.o kbd.o command.o display.o \
          insert.o search.o files.o utils.o

edit : $(objects)
        cc -o edit $(objects)
main.o : defs.h
kbd.o : defs.h command.h
command.o : defs.h command.h
display.o : defs.h buffer.h
insert.o : defs.h buffer.h
search.o : defs.h buffer.h
files.o : defs.h buffer.h command.h
utils.o : defs.h

.PHONY : clean
clean :
        rm edit $(objects)
```

这种方法，也就是 make 的“隐晦规则”。上面文件内容中，“.PHONY”表示，clean 是个伪目标文件。

六、另类风格的 Makefile

即然我们的 make 可以自动推导命令，那么我看到那堆[.o]和[.h]的依赖就有点不爽，那么多的重复的[.h]，能不能把其收拢起来，好吧，没有问题，这个对于 make 来说很容易，谁叫它提供了自动推导命令和文件的功能呢？来看看最新风格的 Makefile:

```
objects = main.o kbd.o command.o display.o \
          insert.o search.o files.o utils.o

edit : $(objects)
        cc -o edit $(objects)
```

```
$(objects) : defs.h
kbd.o command.o files.o : command.h
display.o insert.o search.o files.o : buffer.h

.PHONY : clean
clean :
        rm edit $(objects)
```

这种风格，让我们的 Makefile 变得很简单，但我们的文件依赖关系就显得有点凌乱了。鱼和熊掌不可兼得。还看你的喜好了。我是不喜欢这种风格的，一是文件的依赖关系看不清楚，二是如果文件一多，要加入几个新的.o 文件，那就理不清楚了。

七、清空目标文件的规则

每个 Makefile 中都应该写一个清空目标文件（.o 和执行文件）的规则，这不仅便于重编译，也很利于保持文件的清洁。

```
clean:
    rm edit $(objects)
```

更为稳健的做法是：

```
.PHONY : clean
clean :
        -rm edit $(objects)
```

前面说过，.PHONY 意思表示 clean 是一个“伪目标”，。而在 rm 命令前面加了一个小减号的意思就是，也许某些文件出现问题，但不要管，继续做后面的事。当然，clean 的规则不要放在文件的开头，不然，这就会变成 make 的默认目标，相信谁也不愿意这样。不成文的规矩是——“clean 从来都是放在文件的最后”。

Makefile 总述

一、Makefile 里有什么？

Makefile 里主要包含了五个东西：显式规则、隐晦规则、变量定义、文件指示和注释。

1、显式规则。

显式规则说明了，如何生成一个或多的的目标文件。这是由 Makefile 的书写者明显指出，要生成的文件，文件的依赖文件，生成的命令。

2、隐晦规则。

由于我们的 make 有自动推导的功能，所以隐晦的规则可以让我们比较粗糙地简略地书写 Makefile，这是由 make 所支持的。

3、变量的定义。

在 Makefile 中我们要定义一系列的变量，变量一般都是字符串，这个有点你 C 语言中的宏，

当 Makefile 被执行时，其中的变量都会被扩展到相应的引用位置上。

4、文件指示。

其包括了三个部分，一个是在一个 Makefile 中引用另一个 Makefile，就像 C 语言中的 include 一样；另一个是指根据某些情况指定 Makefile 中的有效部分，就像 C 语言中的预编译 #if 一样；还有就是定义一个多行的命令。有关这一部分的内容，会在后续的部分中讲述。

5、注释。

Makefile 中只有行注释，和 UNIX 的 Shell 脚本一样，其注释是用“#”字符，这个就像 C/C++ 中的“//”一样。如果你要在你的 Makefile 中使用“#”字符，可以用反斜框进行转义，如：“\#”。

最后，还值得一提的是，在 Makefile 中的命令，必须要以[Tab]键开始。

二、Makefile 的文件名

默认的情况下，make 命令会在当前目录下按顺序找寻文件名为“GNUmakefile”、“makefile”、“Makefile”的文件，找到了解释这个文件。在这三个文件名中，最好使用“Makefile”这个文件名，因为，这个文件名第一个字符为大写，这样有一种醒目的感觉。最好不要用“GNUmakefile”，这个文件是 GNU 的 make 识别的。有另外一些 make 只对全小写的“makefile”文件名敏感，但是基本上来说，大多数的 make 都支持“makefile”和“Makefile”这两种默认文件名。

当然，你可以使用别的文件名来书写 Makefile，比如：“Make.Linux”，“Make.Solaris”，“Make.AIX”等，如果要指定特定的 Makefile，你可以使用 make 的“-f”和“--file”参数，如：make -f Make.Linux 或 make --file Make.AIX。

三、引用其它的 Makefile

在 Makefile 使用 include 关键字可以把别的 Makefile 包含进来，这很像 C 语言的#include，被包含的文件会原模原样的放在当前文件的包含位置。include 的语法是：

include <filename>

filename 可以是当前操作系统 Shell 的文件模式（可以保含路径和通配符）

在 include 前面可以有一些空字符，但是绝不能是[Tab]键开始。include 和<filename>可以用一个或多个空格隔开。举个例子，你有这样几个 Makefile：a.mk、b.mk、c.mk，还有一个文件叫 foo.make，以及一个变量$(bar)，其包含了 e.mk 和 f.mk，那么，下面的语句：

include foo.make *.mk $(bar)

等价于：

include foo.make a.mk b.mk c.mk e.mk f.mk

make 命令开始时，会把找寻 include 所指出的其它 Makefile，并把其内容安置在当前的位置。就好像 C/C++的#include 指令一样。如果文件都没有指定绝对路径或是相对路径的话，make 会在当前目录下首先寻找，如果当前目录下没有找到，那么，make 还会在下面的几个目录下找：

1、如果 make 执行时，有“-I”或“--include-dir”参数，那么 make 就会在这个参数所指定的目录下去寻找。

2、如果目录<prefix>/include（一般是：/usr/local/bin 或/usr/include）存在的话，make 也会去找。

如果有文件没有找到的话，make 会生成一条警告信息，但不会马上出现致命错误。它会继续载入其它的文件，一旦完成 Makefile 的读取，make 会再重试这些没有找到，或是不能读取的文件，如果还是不行，make 才会出现一条致命信息。如果你想让 make 不理那些无法读取的文件，而继续执行，你可以在 include 前加一个减号“-”。如：

-include <filename>

其表示，无论 include 过程中出现什么错误，都不要报错继续执行。和其它版本 make 兼容的相关命令是 sinclude，其作用和这一个是一样的。

四、环境变量 MAKEFILES

如果你的当前环境中定义了环境变量 MAKEFILES，那么，make 会把这个变量中的值做一个类似于 include 的动作。这个变量中的值是其它的 Makefile，用空格分隔。只是它和 include 不同的是，从这个环境变中引入的 Makefile 的“目标”不会起作用，如果环境变量中定义的文件发现错误，make 也会不理。

但是在这里我还是建议不要使用这个环境变量，因为只要这个变量一被定义，那么当你使用 make 时，所有的 Makefile 都会受到它的影响，这绝不是你想看到的。在这里提这个事，只是为了告诉大家，也许有时候你的 Makefile 出现了怪事，那么你可以看看当前环境中有没有定义这个变量。

五、make 的工作方式

GNU 的 make 工作时的执行步骤如下：（想来其它的 make 也是类似）

1、读入所有的 Makefile。

2、读入被 include 的其它 Makefile。

3、初始化文件中的变量。

4、推导隐晦规则，并分析所有规则。

5、为所有的目标文件创建依赖关系链。

6、根据依赖关系，决定哪些目标要重新生成。

7、执行生成命令。

1-5 步为第一个阶段，6-7 为第二个阶段。第一个阶段中，如果定义的变量被使用了，那么，make 会把其展开在使用的位置。但 make 并不会完全马上展开，make 使用的是拖延战术，如果变量出现在依赖关系的规则中，那么仅当这条依赖被决定要使用了，变量才会在其内部展开。

当然，这个工作方式你不一定要清楚，但是知道这个方式你也会对 make 更为熟悉。有了这个基础，后续部分也就容易看懂了。

书写规则

规则包含两个部分，一个是依赖关系，一个是生成目标的方法。在 Makefile 中，规则的顺序是很重要的，因为，Makefile 中只应该有一个最终目标，其它的目标都是被这个目标所连带出来的，所以一定要让 make 知道你的最终目标是什么。一般来说，定义在 Makefile 中的目标

可能会有很多，但是第一条规则中的目标将被确立为最终的目标。如果第一条规则中的目标有很多个，那么，第一个目标会成为最终的目标。make 所完成的也就是这个目标。下面，还是让我们来看一看如何书写规则。

一、规则举例

```
foo.o : foo.c defs.h        # foo 模块
        cc -c -g foo.c
```

看到这个例子，各位应该不是很陌生了，前面也已说过，foo.o 是我们的目标，foo.c 和 defs.h 是目标所依赖的源文件，而只有一个命令“cc -c -g foo.c”（以 Tab 键开头）。这个规则告诉我们两件事：

1、文件的依赖关系，foo.o 依赖于 foo.c 和 defs.h 的文件，如果 foo.c 和 defs.h 的文件日期要比 foo.o 文件日期要新，或是 foo.o 不存在，那么依赖关系发生。

2、如果生成（或更新）foo.o 文件。也就是那个 cc 命令，其说明了，如何生成 foo.o 这个文件。（当然 foo.c 文件 include 了 defs.h 文件）

二、规则的语法

```
targets : prerequisites
  command
  ...
```

或是这样：

```
targets : prerequisites ; command
        command
        ...
```

targets 是文件名，以空格分开，可以使用通配符。一般来说，我们的目标基本上是一个文件，但也有可能是多个文件。

command 是命令行，如果其不与“target:prerequisites”在一行，那么，必须以[Tab 键]开头，如果和 prerequisites 在一行，那么可以用分号做为分隔。（见上）

prerequisites 也就是目标所依赖的文件（或依赖目标）。如果其中的某个文件要比目标文件要新，那么，目标就被认为是“过时的”，被认为是需要重生成的。这个在前面已经讲过了。

如果命令太长，你可以使用反斜框（‘\’）作为换行符。make 对一行上有多少个字符没有限制。规则告诉 make 两件事，文件的依赖关系和如何成成目标文件。一般来说，make 会以 UNIX 的标准 Shell，也就是/bin/sh 来执行命令。

三、在规则中使用通配符

如果我们想定义一系列比较类似的文件，我们很自然地就想起使用通配符。make 支持三各通配符：“*”，“?”和“[...]”。这是和 Unix 的 B-Shell 是相同的。

波浪号（“~”）字符在文件名中也有比较特殊的用途。如果是“~/test”，这就表示当前用户的$HOME 目录下的 test 目录。而“~hchen/test”则表示用户 hchen 的宿主目录下的 test 目录。（这些都是 Unix 下的小知识了，make 也支持）而在 Windows 或是 MS-DOS 下，用户没有宿

主目录，那么波浪号所指的目录则根据环境变量“HOME”而定。

通配符代替了你一系列的文件，如“*.c”表示所以后缀为 c 的文件。一个需要我们注意的是，如果我们的文件名中有通配符，如：“*”，那么可以用转义字符“\”，如“*”来表示真实的“*”字符，而不是任意长度的字符串。好吧，还是先来看几个例子吧：

```
clean:
    rm -f *.o
```

上面这个例子我不不多说了，这是操作系统 Shell 所支持的通配符。这是在命令中的通配符。

```
print: *.c
    lpr -p $?
    touch print
```

上面这个例子说明了通配符也可以在我们的规则中，目标 print 依赖于所有的[.c]文件。其中的“$?”是一个自动化变量，我会在后面给你讲述。

```
objects = *.o
```

上面这个例子，表示了，通符同样可以用在变量中。并不是说[*.o]会展开，不！objects 的值就是“*.o”。Makefile 中的变量其实就是 C/C++中的宏。如果你要让通配符在变量中展开，也就是让 objects 的值是所有[.o]的文件名的集合，那么，你可以这样：

```
objects := $(wildcard *.o)
```

这种用法由关键字“wildcard”指出，关于 Makefile 的关键字，我们将在后面讨论。

四、文件搜寻

在一些大的工程中，有大量的源文件，我们通常的做法是把这许多的源文件分类，并存放在不同的目录中。所以，当 make 需要去找寻文件的依赖关系时，你可以在文件前加上路径，但最好的方法是把一个路径告诉 make，让 make 在自动去找。

Makefile 文件中的特殊变量“VPATH”就是完成这个功能的，如果没有指明这个变量，make 只会在当前的目录中去找寻依赖文件和目标文件。如果定义了这个变量，那么，make 就会在当当前目录找不到的情况下，到所指定的目录中去找寻文件了。

```
VPATH = src:../headers
```

上面的的定义指定两个目录，“src”和“../headers”，make 会按照这个顺序进行搜索。目录由“冒号”分隔。（当然，当前目录永远是最高优先搜索的地方）

另一个设置文件搜索路径的方法是使用 make 的“vpath”关键字（注意，它是全小写的），这不是变量，这是一个 make 的关键字，这和上面提到的那个 VPATH 变量很类似，但是它更为灵活。它可以指定不同的文件在不同的搜索目录中。这是一个很灵活的功能。它的使用方法有三种：

1、vpath <pattern> <directories>

为符合模式<pattern>的文件指定搜索目录<directories>。

2、vpath <pattern>

清除符合模式<pattern>的文件的搜索目录。

3、vpath

清除所有已被设置好了的文件搜索目录。

vapth 使用方法中的<pattern>需要包含“%”字符。“%”的意思是匹配零或若干字符，例如，“%.h”表示所有以“.h”结尾的文件。<pattern>指定了要搜索的文件集，而<directories>则指定了<pattern>的文件集的搜索的目录。例如：

vpath %.h ../headers

该语句表示，要求 make 在“../headers”目录下搜索所有以“.h”结尾的文件。（如果某文件在当前目录没有找到的话）我们可以连续地使用 vpath 语句，以指定不同搜索策略。如果连续的 vpath 语句中出现了相同的<pattern>，或是被重复了的<pattern>，那么，make 会按照 vpath 语句的先后顺序来执行搜索。如：

vpath %.c foo

vpath % blish

vpath %.c bar

其表示“.c”结尾的文件，先在“foo”目录，然后是“blish”，最后是“bar”目录。

vpath %.c foo:bar

vpath % blish

而上面的语句则表示“.c”结尾的文件，先在“foo”目录，然后是“bar”目录，最后才是“blish”目录。

五、伪目标

最早先的一个例子中，我们提到过一个“clean”的目标，这是一个“伪目标”，

clean:

```
        rm *.o temp
```

正像我们前面例子中的“clean”一样，既然我们生成了许多文件编译文件，我们也应该提供一个清除它们的“目标”以备完整地重编译而用。（以“make clean”来使用该目标）因为，我们并不生成“clean”这个文件。“伪目标”并不是一个文件，只是一个标签，由于“伪目标”不是文件，所以 make 无法生成它的依赖关系和决定它是否要执行。我们只有通过显示地指明这个“目标”才能让其生效。当然，“伪目标”的取名不能和文件名重名，不然其就失去了“伪目标”的意义了。当然，为了避免和文件重名的这种情况，我们可以使用一个特殊的标记“.PHONY”来显示地指明一个目标是“伪目标”，向 make 说明，不管是否有这个文件，这个目标就是“伪目标”。

.PHONY : clean

只要有这个声明，不管是否有“clean”文件，要运行“clean”这个目标，只有“make clean”这样。于是整个过程可以这样写：

```
 .PHONY: clean
clean:
```

```
        rm *.o temp
```

伪目标一般没有依赖的文件。但是，我们也可以为伪目标指定所依赖的文件。伪目标同样可以作为“默认目标”，只要将其放在第一个。一个示例就是，如果你的 Makefile 需要一口气生成若干个可执行文件，但你只想简单地敲一个 make 完事，并且，所有的目标文件都写在一个 Makefile 中，那么你可以使用“伪目标”这个特性：

```
all : prog1 prog2 prog3
.PHONY : all

prog1 : prog1.o utils.o
        cc -o prog1 prog1.o utils.o
prog2 : prog2.o
        cc -o prog2 prog2.o
prog3 : prog3.o sort.o utils.o
        cc -o prog3 prog3.o sort.o utils.o
```

我们知道，Makefile 中的第一个目标会被作为其默认目标。我们声明了一个“all”的伪目标，其依赖于其它三个目标。由于伪目标的特性是，总是被执行的，所以其依赖的那三个目标就总是不如“all”这个目标新。所以，其它三个目标的规则总是会被决议。也就达到了我们一口气生成多个目标的目的。“.PHONY : all”声明了“all”这个目标为“伪目标”。

随便提一句，从上面的例子我们可以看出，目标也可以成为依赖。所以，伪目标同样也可成为依赖。看下面的例子：

```
.PHONY: cleanall cleanobj cleandiff

cleanall : cleanobj cleandiff
        rm program

cleanobj :
        rm *.o

cleandiff :
        rm *.diff
```

“make clean”将清除所有要被清除的文件。“cleanobj”和“cleandiff”这两个伪目标有点像“子程序”的意思。我们可以输入“make cleanall”和“make cleanobj”和“make cleandiff”命令来达到清除不同种类文件的目的。

六、多目标

Makefile 的规则中的目标可以不止一个，其支持多目标，有可能我们的多个目标同时依赖于一个文件，并且其生成的命令大体类似。于是我们就能把其合并起来。当然，多个目标的生

成规则的执行命令是同一个，这可能会可我们带来麻烦，不过好在我们的可以使用一个自动化变量“$@”(关于自动化变量，将在后面讲述)，这个变量表示着目前规则中所有的目标的集合，这样说可能很抽象，还是看一个例子吧。

```
bigoutput littleoutput : text.g
        generate text.g -$(subst output,,$@) > $@
```

上述规则等价于：

```
bigoutput : text.g
        generate text.g -big > bigoutput
littleoutput : text.g
        generate text.g -little > littleoutput
```

其中，-$(subst output,,$@)中的“$”表示执行一个 Makefile 的函数，函数名为 subst，后面的为参数。关于函数，将在后面讲述。这里的这个函数是截取字符串的意思，“$@”表示目标的集合，就像一个数组，“$@”依次取出目标，并执于命令。

七、静态模式

静态模式可以更加容易地定义多目标的规则，可以让我们的规则变得更加的有弹性和灵活。我们还是先来看一下语法：

```
<targets ...>: <target-pattern>: <prereq-patterns ...>
        <commands>
        ...
```

targets 定义了一系列的目标文件，可以有通配符。是目标的一个集合。

target-parrtern 是指明了 targets 的模式，也就是的目标集模式。

prereq-parrterns 是目标的依赖模式，它对 target-parrtern 形成的模式再进行一次依赖目标的定义。

这样描述这三个东西，可能还是没有说清楚，还是举个例子来说明一下吧。如果我们的<target-parrtern>定义成“%.o”，意思是我们的<target>集合中都是以“.o”结尾的，而如果我们的<prereq-parrterns>定义成“%.c”，意思是对<target-parrtern>所形成的目标集进行二次定义，其计算方法是，取<target-parrtern>模式中的“%”(也就是去掉了[.o]这个结尾)，并为其加上[.c]这个结尾，形成的新集合。

所以，我们的“目标模式”或是“依赖模式”中都应该有“%”这个字符，如果你的文件名中有“%”那么你可以使用反斜杠“\”进行转义，来标明真实的“%”字符。

看一个例子：

```
objects = foo.o bar.o

all: $(objects)
$(objects): %.o: %.c
        $(CC) -c $(CFLAGS) $< -o $@
```

上面的例子中，指明了我们的目标从$object 中获取，“%.o”表明要所有以“.o”结尾的目标，也就是“foo.o bar.o”，也就是变量$object 集合的模式，而依赖模式“%.c”则取模式“%.o”的“%”，也就是“foo bar”，并为其加下“.c”的后缀，于是，我们的依赖目标就是“foo.c bar.c”。而命令中的“$<”和“$@”则是自动化变量，“$<”表示所有的依赖目标集（也就是“foo.c bar.c”），“$@”表示目标集（也就是“foo.o bar.o”）。于是，上面的规则展开后等价于下面的规则：

```
foo.o : foo.c
        $(CC) -c $(CFLAGS) foo.c -o foo.o
bar.o : bar.c
        $(CC) -c $(CFLAGS) bar.c -o bar.o
```

试想，如果我们的“%.o”有几百个，那种我们只要用这种很简单的“静态模式规则”就可以写完一堆规则，实在是太有效率了。“静态模式规则”的用法很灵活，如果用得好，那会一个很强大的功能。再看一个例子：

```
files = foo.elc bar.o lose.o

$(filter %.o,$(files)): %.o: %.c
        $(CC) -c $(CFLAGS) $< -o $@
$(filter %.elc,$(files)): %.elc: %.el
        emacs -f batch-byte-compile $<
```

$(filter %.o,$(files))表示调用 Makefile 的 filter 函数，过滤“$filter”集，只要其中模式为“%.o”的内容。其的它内容，我就不用多说了吧。这个例字展示了 Makefile 中更大的弹性。

八、自动生成依赖性

在 Makefile 中，我们的依赖关系可能会需要包含一系列的头文件，比如，如果我们的 main.c 中有一句“#include "defs.h"”，那么我们的依赖关系应该是：

```
main.o : main.c defs.h
```

但是，如果是一个比较大型的工程，你必须清楚哪些 C 文件包含了哪些头文件，并且，你在加入或删除头文件时，也需要小心地修改 Makefile，这是一个很没有维护性的工作。为了避免这种繁重而又容易出错的事情，我们可以使用 C/C++编译的一个功能。大多数的 C/C++编译器都支持一个“-M”的选项，即自动找寻源文件中包含的头文件，并生成一个依赖关系。例如，如果我们执行下面的命令：

```
cc -M main.c
```

其输出是：

```
main.o : main.c defs.h
```

于是由编译器自动生成的依赖关系，这样一来，你就不必再手动书写若干文件的依赖关系，而由编译器自动生成了。需要提醒一句的是，如果你使用 GNU 的 C/C++编译器，你得用“-MM”

参数，不然，“-M”参数会把一些标准库的头文件也包含进来。

gcc -M main.c 的输出是：

```
main.o: main.c defs.h /usr/include/stdio.h /usr/include/features.h \
     /usr/include/sys/cdefs.h /usr/include/gnu/stubs.h \
     /usr/lib/gcc-lib/i486-suse-linux/2.95.3/include/stddef.h \
     /usr/include/bits/types.h /usr/include/bits/pthreadtypes.h \
     /usr/include/bits/sched.h /usr/include/libio.h \
     /usr/include/_G_config.h /usr/include/wchar.h \
     /usr/include/bits/wchar.h /usr/include/gconv.h \
     /usr/lib/gcc-lib/i486-suse-linux/2.95.3/include/stdarg.h \
     /usr/include/bits/stdio_lim.h
```

gcc -MM main.c 的输出则是：

```
main.o: main.c defs.h
```

那么，编译器的这个功能如何与我们的 Makefile 联系在一起呢。因为这样一来，我们的 Makefile 也要根据这些源文件重新生成，让 Makefile 自己依赖于源文件？这个功能并不现实，不过我们可以有其它手段来迂回地实现这一功能。GNU 组织建议把编译器为每一个源文件的自动生成的依赖关系放到一个文件中，为每一个“name.c”的文件都生成一个“name.d”的 Makefile 文件，[.d]文件中就存放对应[.c]文件的依赖关系。于是，我们可以写出[.c]文件和[.d]文件的依赖关系，并让 make 自动更新或自成[.d]文件，并把其包含在我们的主 Makefile 中，这样，我们就可以自动化地生成每个文件的依赖关系了。这里，我们给出了一个模式规则来产生[.d]文件：

```
%.d: %.c
        @set -e; rm -f $@; \
         $(CC) -M $(CPPFLAGS) $< > $@.$$$$; \
         sed 's,\($*\)\.o[ :]*,\1.o $@ : ,g' < $@.$$$$ > $@; \
         rm -f $@.$$$$
```

这个规则的意思是，所有的[.d]文件依赖于[.c]文件，“rm -f $@”的意思是删除所有的目标，也就是[.d]文件，第二行的意思是，为每个依赖文件“$<”，也就是[.c]文件生成依赖文件，“$@”表示模式“%.d”文件，如果有一个 C 文件是 name.c，那么“%”就是“name”，“$$$$”意为一个随机编号，第二行生成的文件有可能是“name.d.12345”，第三行使用 sed 命令做了一个替换，关于 sed 命令的用法请参看相关的使用文档。第四行就是删除临时文件。总而言之，这个模式要做的事就是在编译器生成的依赖关系中加入[.d]文件的依赖，即把依赖关系：

```
main.o : main.c defs.h
```

转成：

```
main.o main.d : main.c defs.h
```

于是，我们的[.d]文件也会自动更新了，并会自动生成了，当然，你还可以在这个[.d]文件中加入的不只是依赖关系，包括生成的命令也可一并加入，让每个[.d]文件都包含一个完赖的规则。一旦我们完成这个工作，接下来，我们就要把这些自动生成的规则放进我们的主 Makefile 中。我们可以使用 Makefile 的“include”命令，来引入别的 Makefile 文件（前面讲过），例如：

```
sources = foo.c bar.c
include $(sources:.c=.d)
```

上述语句中的“$(sources:.c=.d)”中的“.c=.d”的意思是做一个替换，把变量$(sources)所有[.c]的字串都替换成[.d]，关于这个“替换”的内容，在后面我会有更为详细的讲述。当然，你得注意次序，因为 include 是按次来载入文件，最先载入的[.d]文件中的目标会成为默认目标。

书写命令

每条规则中的命令和操作系统 Shell 的命令行是一致的。make 会按顺序一条一条的执行命令，每条命令的开头必须以[Tab]键开头。除非命令是紧跟在依赖规则后面的分号后的。在命令行之间中的空格或是空行会被忽略，但是如果该空格或空行是以 Tab 键开头的，那么 make 会认为其是一个空命令。

我们在 UNIX 下可能会使用不同的 Shell，但是 make 的命令默认是被“/bin/sh”——UNIX 的标准 Shell 解释执行的。除非你特别指定一个其它的 Shell。Makefile 中，“#”是注释符，很像 C/C++中的“//”，其后的本行字符都被注释。

一、显示命令

通常，make 会把其要执行的命令行在命令执行前输出到屏幕上。当我们用“@”字符在命令行前，那么，这个命令将不被 make 显示出来，最具代表性的例子是，我们用这个功能来像屏幕显示一些信息。如：

```
@echo 正在编译 XXX 模块......
```

当 make 执行时，会输出“正在编译 XXX 模块......”字串，但不会输出命令，如果没有“@”，那么，make 将输出：

```
echo 正在编译 XXX 模块......
正在编译 XXX 模块......
```

如果 make 执行时，带入 make 参数“-n”或“--just-print”，那么其只是显示命令，但不会执行命令，这个功能很有利于我们调试我们的 Makefile，看看我们书写的命令是执行起来是什么样子的或是什么顺序的。而 make 参数“-s”或“--slient”则是全面禁止命令的显示。

二、命令执行

当依赖目标新于目标时，也就是当规则的目标需要被更新时，make 会一条一条的执行其后的命令。需要注意的是，如果你要让上一条命令的结果应用在下一条命令时，你应该使用分号分隔这两条命令。比如你的第一条命令是 cd 命令，你希望第二条命令得在 cd 之后的基础上运行，那么你就不能把这两条命令写在两行上，而应该把这两条命令写在一行上，用分号分隔。

如：

示例一：

```
exec:
        cd /home/hchen
        pwd
```

示例二：

```
exec:
        cd /home/hchen; pwd
```

当我们执行“make exec”时，第一个例子中的 cd 没有作用，pwd 会打印出当前的 Makefile 目录，而第二个例子中，cd 就起作用了，pwd 会打印出“/home/hchen”。make 一般是使用环境变量 SHELL 中所定义的系统 Shell 来执行命令,默认情况下使用 UNIX 的标准 Shell——/bin/sh 来执行命令。但在 MS-DOS 下有点特殊，因为 MS-DOS 下没有 SHELL 环境变量，当然你也可以指定。如果你指定了 UNIX 风格的目录形式，首先，make 会在 SHELL 所指定的路径中找寻命令解释器，如果找不到，其会在当前盘符中的当前目录中寻找，如果再找不到，其会在 PATH 环境变量中所定义的所有路径中寻找。MS-DOS 中，如果你定义的命令解释器没有找到，其会给你的命令解释器加上诸如“.exe”、“.com”、“.bat”、“.sh”等后缀。

三、命令出错

每当命令运行完后，make 会检测每个命令的返回码，如果命令返回成功，那么 make 会执行下一条命令，当规则中所有的命令成功返回后，这个规则就算是成功完成了。如果一个规则中的某个命令出错了（命令退出码非零），那么 make 就会终止执行当前规则，这将有可能终止所有规则的执行。

有些时候，命令的出错并不表示就是错误的。例如 mkdir 命令，我们一定需要建立一个目录，如果目录不存在，那么 mkdir 就成功执行，万事大吉，如果目录存在，那么就出错了。我们之所以使用 mkdir 的意思就是一定要有这样的一个目录，于是我们就不希望 mkdir 出错而终止规则的运行。

为了做到这一点，忽略命令的出错，我们可以在 Makefile 的命令行前加一个减号“-”（在 Tab 键之后），标记为不管命令出不出错都认为是成功的。如：

```
clean:
        -rm -f *.o
```

还有一个全局的办法是，给 make 加上“-i”或是“--ignore-errors”参数，那么，Makefile 中所有命令都会忽略错误。而如果一个规则是以“.IGNORE”作为目标的，那么这个规则中的所有命令将会忽略错误。这些是不同级别的防止命令出错的方法，你可以根据你的不同喜爱进行设置。还有一个要提一下的 make 的参数的是“-k”或是“--keep-going”，这个参数的意思是，如果某规则中的命令出错了，那么就终目该规则的执行，但继续执行其它规则。

四、嵌套执行 make

在一些大的工程中，我们会把我们不同模块或是不同功能的源文件放在不同的目录中，我

们可以在每个目录中都书写一个该目录的 Makefile，这有利于让我们的 Makefile 变得更加地简洁，而不至于把所有的东西全部写在一个 Makefile 中，这样会很难维护我们的 Makefile，这个技术对于我们模块编译和分段编译有着非常大的好处。例如，我们有一个子目录叫 subdir，这个目录下有个 Makefile 文件，来指明了这个目录下文件的编译规则。那么我们总控的 Makefile 可以这样书写：

```
subsystem:
        cd subdir && $(MAKE)
```

其等价于：

```
subsystem:
        $(MAKE) -C subdir
```

定义$(MAKE)宏变量的意思是，也许我们的 make 需要一些参数，所以定义成一个变量比较利于维护。这两个例子的意思都是先进入“subdir”目录，然后执行 make 命令。我们把这个 Makefile 叫做“总控 Makefile”，总控 Makefile 的变量可以传递到下级的 Makefile 中（如果你显示的声明），但是不会覆盖下层的 Makefile 中所定义的变量，除非指定了“-e”参数。如果你要传递变量到下级 Makefile 中，那么你可以使用这样的声明：

```
export <variable ...>
```

如果你不想让某些变量传递到下级 Makefile 中，那么你可以这样声明：

```
unexport <variable ...>
```

如：示例一：

```
    export variable = value
```

其等价于：

```
    variable = value
    export variable
```

其等价于：

```
    export variable := value
```

其等价于：

```
    variable := value
    export variable
```

示例二：

```
    export variable += value
```

其等价于：

```
    variable += value
    export variable
```

如果你要传递所有的变量，那么，只要一个 export 就行了。后面什么也不用跟，表示传递所有的变量。需要注意的是，有两个变量，一个是 SHELL，一个是 MAKEFLAGS，这两个变量不管你是否 export，其总是要传递到下层 Makefile 中，特别是 MAKEFILES 变量，其中包含了 make

的参数信息，如果我们执行“总控 Makefile”时有 make 参数或是在上层 Makefile 中定义了这个变量，那么 MAKEFILES 变量将会是这些参数，并会传递到下层 Makefile 中，这是一个系统级的环境变量。但是 make 命令中的有几个参数并不往下传递，它们是“-C”，“-f”，“-h”“-o”和“-W”（有关 Makefile 参数的细节将在后面说明），如果你不想往下层传递参数，那么，你可以这样来：

```
subsystem:
        cd subdir && $(MAKE) MAKEFLAGS=
```

如果你定义了环境变量 MAKEFLAGS，那么你得确信其中的选项是大家都会用到的，如果其中有“-t”，“-n”,和“-q”参数，那么将会有让你意想不到的结果，或许会让你异常地恐慌。还有一个在“嵌套执行”中比较有用的参数，“-w”或是“--print-directory”会在 make 的过程中输出一些信息，让你看到目前的工作目录。比如，如果我们的下级 make 目录是“/home/hchen/gnu/make”，如果我们使用“make -w”来执行，那么当进入该目录时，我们会看到：

```
make: Entering directory `/home/hchen/gnu/make'.
```

而在完成下层 make 后离开目录时，我们会看到：

```
make: Leaving directory `/home/hchen/gnu/make'
```

当你使用“-C”参数来指定 make 下层 Makefile 时，“-w”会被自动打开的。如果参数中有“-s”（“--slient”）或是“--no-print-directory”，那么，“-w”总是失效的。

五、定义命令包

如果 Makefile 中出现一些相同命令序列，那么我们可以为这些相同的命令序列定义一个变量。定义这种命令序列的语法以“define”开始，以“endef”结束，如：

```
define run-yacc
yacc $(firstword $^)
mv y.tab.c $@
endef
```

这里，“run-yacc”是这个命令包的名字，其不要和 Makefile 中的变量重名。在“define”和“endef”中的两行就是命令序列。这个命令包中的第一个命令是运行 Yacc 程序，因为 Yacc 程序总是生成“y.tab.c”的文件，所以第二行的命令就是把这个文件改改名字。还是把这个命令包放到一个示例中来看看吧。

```
foo.c : foo.y
        $(run-yacc)
```

我们可以看见，要使用这个命令包，我们就好像使用变量一样。在这个命令包的使用中，命令包“run-yacc”中的“$^”就是“foo.y”，“$@”就是“foo.c”（有关这种以“$”开头的特殊变量，我们会在后面介绍），make 在执行命令包时，命令包中的每个命令会被依次独立执行。

使用变量

在 Makefile 中的定义的变量，就像是 C/C++语言中的宏一样，他代表了一个文本字串，在 Makefile 中执行的时候其会自动原模原样地展开在所使用的地方。其与 C/C++所不同的是，你可以在 Makefile 中改变其值。在 Makefile 中，变量可以使用在“目标”，“依赖目标”，“命令”或是 Makefile 的其它部分中。

变量的命名字可以包含字符、数字，下划线（可以是数字开头），但不应该含有“:”、“#”、“=”或是空字符（空格、回车等）。变量是大小写敏感的，“foo”、“Foo”和“FOO”是三个不同的变量名。传统的 Makefile 的变量名是全大写的命名方式，但我推荐使用大小写搭配的变量名，如：MakeFlags。这样可以避免和系统的变量冲突，而发生意外的事情。有一些变量是很奇怪字串，如“$<”、“$@”等，这些是自动化变量，这些会在后面介绍。

一、变量的基础

变量在声明时需要给予初值，而在使用时，需要给在变量名前加上“$”符号，但最好用小括号“()”或是大括号“{}”把变量给包括起来。如果你要使用真实的“$”字符，那么你需要用“$$”来表示。变量可以使用在许多地方，如规则中的“目标”、“依赖”、“命令”以及新的变量中。先看一个例子：

```
objects = program.o foo.o utils.o
program : $(objects)
        cc -o program $(objects)

$(objects) : defs.h
```

变量会在使用它的地方精确地展开，就像 C/C++中的宏一样，例如：

```
foo = c
prog.o : prog.$(foo)
        $(foo)$(foo) -$(foo) prog.$(foo)
```

展开后得到：

```
prog.o : prog.c
        cc -c prog.c
```

当然，千万不要在你的 Makefile 中这样干，这里只是举个例子来表明 Makefile 中的变量在使用处展开的真实样子。可见其就是一个“替代”的原理。另外，给变量加上括号完全是为了更加安全地使用这个变量，在上面的例子中，如果你不想给变量加上括号，那也可以，但我还是强烈建议你给变量加上括号。

二、变量中的变量

在定义变量的值时，我们可以使用其它变量来构造变量的值，在 Makefile 中有两种方式来在用变量定义变量的值。先看第一种方式，也就是简单的使用“=”号，在“=”左侧是变量，右侧是变量的值，右侧变量的值可以定义在文件的任何一处，也就是说，右侧中的变量不一定非要是已定义好的值，其也可以使用后面定义的值。如：

```
foo = $(bar)
```

```
bar = $(ugh)
ugh = Huh?
all:
        echo $(foo)
```

我们执行“make all”将会打出变量$(foo)的值是“Huh?”($(foo)的值是$(bar)，$(bar)的值是$(ugh)，$(ugh)的值是“Huh?”) 可见，变量是可以使用后面的变量来定义的。这个功能有好的地方，也有不好的地方，好的地方是，我们可以把变量的真实值推到后面来定义，如：

```
CFLAGS = $(include_dirs) -O
include_dirs = -Ifoo -Ibar
```

当“CFLAGS”在命令中被展开时，会是“-Ifoo -Ibar -O”。但这种形式也有不好的地方，那就是递归定义，如：

```
CFLAGS = $(CFLAGS) -O
```

或：

```
A = $(B)
B = $(A)
```

这会让 make 陷入无限的变量展开过程中去，当然，我们的 make 是有能力检测这样的定义，并会报错。还有就是如果在变量中使用函数，那么，这种方式会让我们的 make 运行时非常慢，更糟糕的是，他会使用得两个 make 的函数“wildcard”和“shell”发生不可预知的错误。因为你不会知道这两个函数会被调用多少次。为了避免上面的这种方法，我们可以使用 make 中的另一种用变量来定义变量的方法。这种方法使用的是“:=”操作符，如：

```
x := foo
y := $(x) bar
x := later
```

其等价于：

```
y := foo bar
x := later
```

值得一提的是，这种方法，前面的变量不能使用后面的变量，只能使用前面已定义好了的变量。如果是这样：

```
y := $(x) bar
x := foo
```

那么，y 的值是“bar”，而不是“foo bar”。

上面都是一些比较简单的变量使用了，让我们来看一个复杂的例子，其中包括了 make 的函数、条件表达式和一个系统变量“MAKELEVEL”的使用：

```
ifeq (0,${MAKELEVEL})
cur-dir   := $(shell pwd)
whoami    := $(shell whoami)
```

```
host-type := $(shell arch)
MAKE := ${MAKE} host-type=${host-type} whoami=${whoami}
endif
```

关于条件表达式和函数，我们在后面再说，对于系统变量“MAKELEVEL”，其意思是，如果我们的 make 有一个嵌套执行的动作（参见前面的“嵌套使用 make”），那么，这个变量会记录了我们的当前 Makefile 的调用层数。

下面再介绍两个定义变量时我们需要知道的，请先看一个例子，如果我们要定义一个变量，其值是一个空格，那么我们可以这样来：

```
nullstring :=
space := $(nullstring) # end of the line
```

nullstring 是一个 Empty 变量，其中什么也没有，而我们的 space 的值是一个空格。因为在操作符的右边是很难描述一个空格的，这里采用的技术很管用，先用一个 Empty 变量来标明变量的值开始了，而后面采用“#”注释符来表示变量定义的终止，这样，我们可以定义出其值是一个空格的变量。请注意这里关于“#”的使用，注释符“#”的这种特性值得我们注意，如果我们这样定义一个变量：

```
dir := /foo/bar    # directory to put the frobs in
```

dir 这个变量的值是“/foo/bar”，后面还跟了 4 个空格，如果我们这样使用这样变量来指定别的目录——“$(dir)/file”那么就完蛋了。还有一个比较有用的操作符是“?=”，先看示例：

```
FOO ?= bar
```

其含义是，如果 FOO 没有被定义过，那么变量 FOO 的值就是“bar”，如果 FOO 先前被定义过，那么这条语将什么也不做，其等价于：

```
ifeq ($(origin FOO), undefined)
  FOO = bar
Endif
```

三、变量高级用法

这里介绍两种变量的高级使用方法，第一种是变量值的替换。我们可以替换变量中的共有的部分，其格式是“$(var:a=b)”或是“${var:a=b}”，其意思是，把变量“var”中所有以“a”字串“结尾”的“a”替换成“b”字串。这里的“结尾”意思是“空格”或是“结束符”。还是看一个示例吧：

```
foo := a.o b.o c.o
bar := $(foo:.o=.c)
```

这个示例中，我们先定义了一个“$(foo)”变量，而第二行的意思是把“$(foo)”中所有以“.o”字串“结尾”全部替换成“.c”，所以我们的“$(bar)”的值就是“a.c b.c c.c”。另外一种变量替换的技术是以“静态模式”（参见前面章节）定义的，如：

```
foo := a.o b.o c.o
```

```
bar := $(foo:%.o=%.c)
```

这依赖于被替换字串中的有相同的模式，模式中必须包含一个“%”字符，这个例子同样让$(bar)变量的值为“a.c b.c c.c”。

第二种高级用法是——“把变量的值再当成变量”。先看一个例子：

```
x = y
y = z
a := $($(x))
```

在这个例子中，$(x)的值是“y”，所以$($(x))就是$(y)，于是$(a)的值就是“z”。（注意，是“x=y”，而不是“x=$(y)”）

我们还可以使用更多的层次：

```
x = y
y = z
z = u
a := $($($(x)))
```

这里的$(a)的值是“u”，相关的推导留给读者自己去做吧。让我们再复杂一点，使用上“在变量定义中使用变量”的第一个方式，来看一个例子：

```
x = $(y)
y = z
z = Hello
a := $($(x))
```

这里的$($(x))被替换成了$($(y))，因为$(y)值是“z”，所以，最终结果是：a:=$(z)，也就是“Hello”。再复杂一点，我们再加上函数：

```
x = variable1
variable2 := Hello
y = $(subst 1,2,$(x))
z = y
a := $($($(z)))
```

这个例子中,“$($($(z)))”扩展为“$($(y))”,而其再次被扩展为“$($(subst 1,2,$(x)))”。$(x)的值是“variable1”，subst 函数把“variable1”中的所有“1”字串替换成“2”字串，于是，“variable1”变成“variable2”，再取其值，所以，最终，$(a)的值就是$(variable2)的值——“Hello”。在这种方式中，或要可以使用多个变量来组成一个变量的名字，然后再取其值：

```
first_second = Hello
a = first
b = second
all = $($a_$b)
```

这里的“$a_$b”组成了“first_second”，于是，$(all)的值就是“Hello”。再来看看结合第一种技术的例子：

```
a_objects := a.o b.o c.o
1_objects := 1.o 2.o 3.o
sources := $($(a1)_objects:.o=.c)
```

这个例子中，如果$(a1)的值是“a”的话，那么，$(sources)的值就是“a.c b.c c.c”；如果$(a1)的值是“1”，那么$(sources)的值是“1.c 2.c 3.c”。再来看一个这种技术和“函数”与“条件语句”一同使用的例子：

```
ifdef do_sort
func := sort
else
func := strip
endif

bar := a d b g q c

foo := $($(func) $(bar))
```

这个示例中，如果定义了“do_sort”，那么：foo := $(sort a d b g q c)，于是$(foo)的值就是“a b c d g q”，而如果没有定义“do_sort”，那么：foo := $(sort a d b g q c)，调用的就是 strip 函数。当然，“把变量的值再当成变量”这种技术，同样可以用在操作符的左边：

```
dir = foo
$(dir)_sources := $(wildcard $(dir)/*.c)
define $(dir)_print
lpr $($(dir)_sources)
endef
```

这个例子中定义了三个变量：“dir”，“foo_sources”和“foo_print”。

四、追加变量值

我们可以使用“+=”操作符给变量追加值，如：

```
objects = main.o foo.o bar.o utils.o
objects += another.o
```

于是，我们的$(objects)值变成：“main.o foo.o bar.o utils.o another.o”（another.o 被追加进去了）使用“+=”操作符，可以模拟为下面的这种例子：

```
objects = main.o foo.o bar.o utils.o
objects := $(objects) another.o
```

所不同的是，用“+=”更为简洁。如果变量之前没有定义过，那么，“+=”会自动变成“=”，

如果前面有变量定义，那么“+=”会继承于前次操作的赋值符。如果前一次的是“:=”，那么“+=”会以“:=”作为其赋值符，如：

```
variable := value
variable += more
```

等价于：

```
variable := value
variable := $(variable) more
```

但如果是这种情况：

```
variable = value
variable += more
```

由于前次的赋值符是“=”，所以“+=”也会以“=”来做为赋值，那么岂不会发生变量的递补归定义，这是很不好的，所以 make 会自动为我们解决这个问题，我们不必担心这个问题。

五、override 指示符

如果有变量是通常 make 的命令行参数设置的，那么 Makefile 中对这个变量的赋值会被忽略。如果你想在 Makefile 中设置这类参数的值，那么，你可以使用“override”指示符。其语法是：

```
override <variable> = <value>
override <variable> := <value>
```

当然，你还可以追加：

```
override <variable> += <more text>
```

对于多行的变量定义，我们用 define 指示符，在 define 指示符前，也同样可以使用 ovveride 指示符，如：

```
override define foo
bar
endef
```

六、多行变量

还有一种设置变量值的方法是使用 define 关键字。使用 define 关键字设置变量的值可以有换行，这有利于定义一系列的命令（前面我们讲过“命令包”的技术就是利用这个关键字）。define 指示符后面跟的是变量的名字，而重起一行定义变量的值，定义是以 endef 关键字结束。其工作方式和“=”操作符一样。变量的值可以包含函数、命令、文字，或是其它变量。因为命令需要以[Tab]键开头，所以如果你用 define 定义的命令变量中没有以[Tab]键开头，那么 make 就不会把其认为是命令。下面的这个示例展示了 define 的用法：

```
define two-lines
echo foo
echo $(bar)
endef
```

七、环境变量

make 运行时的系统环境变量可以在 make 开始运行时被载入到 Makefile 文件中，但是如果 Makefile 中已定义了这个变量，或是这个变量由 make 命令行带入，那么系统的环境变量的值将被覆盖。(如果 make 指定了“-e”参数，那么，系统环境变量将覆盖 Makefile 中定义的变量)因此，如果我们在环境变量中设置了“CFLAGS”环境变量，那么我们就可以在所有的 Makefile 中使用这个变量了。这对于我们使用统一的编译参数有比较大的好处。如果 Makefile 中定义了 CFLAGS，那么则会使用 Makefile 中的这个变量，如果没有定义则使用系统环境变量的值，一个共性和个性的统一，很像“全局变量”和“局部变量”的特性。

当 make 嵌套调用时(参见前面的“嵌套调用”章节)，上层 Makefile 中定义的变量会以系统环境变量的方式传递到下层的 Makefile 中。当然，默认情况下，只有通过命令行设置的变量会被传递。而定义在文件中的变量，如果要向下层 Makefile 传递，则需要使用 exprot 关键字来声明。(参见前面章节)。当然，我并不推荐把许多的变量都定义在系统环境中，这样，在我们执行不用的 Makefile 时，拥有的是同一套系统变量，这可能会带来更多的麻烦。

八、目标变量

前面我们所讲的在 Makefile 中定义的变量都是“全局变量”，在整个文件，我们都可以访问这些变量。当然，“自动化变量”除外，如“$<”等这种类量的自动化变量就属于“规则型变量”，这种变量的值依赖于规则的目标和依赖目标的定义。

当然，我样同样可以为某个目标设置局部变量，这种变量被称为“Target-specific Variable”，它可以和“全局变量”同名，因为它的作用范围只在这条规则以及连带规则中，所以其值也只在作用范围内有效。而不会影响规则链以外的全局变量的值。

其语法是：

<target ...> : <variable-assignment>

<target ...> : overide <variable-assignment>

<variable-assignment>可以是前面讲过的各种赋值表达式，如“=”、“:=”、“+=”或是“?=”。第二个语法是针对于 make 命令行带入的变量，或是系统环境变量。这个特性非常的有用，当我们设置了这样一个变量，这个变量会作用到由这个目标所引发的所有的规则中去。如：

```
prog : CFLAGS = -g
prog : prog.o foo.o bar.o
        $(CC) $(CFLAGS) prog.o foo.o bar.o

prog.o : prog.c
        $(CC) $(CFLAGS) prog.c

foo.o : foo.c
        $(CC) $(CFLAGS) foo.c
```

```
bar.o : bar.c
        $(CC) $(CFLAGS) bar.c
```

在这个示例中，不管全局的$(CFLAGS)的值是什么，在 prog 目标，以及其所引发的所有规则中（prog.o foo.o bar.o 的规则），$(CFLAGS)的值都是“-g”

九、模式变量

在 GNU 的 make 中，还支持模式变量（Pattern-specific Variable），通过上面的目标变量中，我们知道，变量可以定义在某个目标上。模式变量的好处就是，我们可以给定一种“模式”，可以把变量定义在符合这种模式的所有目标上。我们知道，make 的“模式”一般是至少含有一个“%”的，所以，我们可以以如下方式给所有以[.o]结尾的目标定义目标变量：

```
%.o : CFLAGS = -O
```

同样，模式变量的语法和“目标变量”一样：

```
<pattern ...> : <variable-assignment>
<pattern ...> : override <variable-assignment>
```

override 同样是针对于系统环境传入的变量，或是 make 命令行指定的变量。

使用条件判断

使用条件判断，可以让 make 根据运行时的不同情况选择不同的执行分支。条件表达式可以是比较变量的值，或是比较变量和常量的值。

一、示例

下面的例子，判断$(CC)变量是否“gcc”，如果是的话，则使用 GNU 函数编译目标。

```
libs_for_gcc = -lgnu
normal_libs =

foo: $(objects)
ifeq ($(CC),gcc)
        $(CC) -o foo $(objects) $(libs_for_gcc)
else
        $(CC) -o foo $(objects) $(normal_libs)
endif
```

可见，在上面示例的这个规则中，目标“foo”可以根据变量“$(CC)”值来选取不同的函数库来编译程序。我们可以从上面的示例中看到三个关键字：ifeq、else 和 endif。ifeq 的意思表示条件语句的开始，并指定一个条件表达式，表达式包含两个参数，以逗号分隔，表达式以圆括号括起。else 表示条件表达式为假的情况。endif 表示一个条件语句的结束，任何一个条件表达式都应该以 endif 结束。当我们的变量$(CC)值是“gcc”时，目标 foo 的规则是：

```
foo: $(objects)
        $(CC) -o foo $(objects) $(libs_for_gcc)
```

而当我们的变量$(CC)值不是“gcc”时（比如“cc”），目标 foo 的规则是：

```
foo: $(objects)
        $(CC) -o foo $(objects) $(normal_libs)
```

当然，我们还可以把上面的那个例子写得更简洁一些：

```
libs_for_gcc = -lgnu
normal_libs =

ifeq ($(CC),gcc)
  libs=$(libs_for_gcc)
else
  libs=$(normal_libs)
endif

foo: $(objects)
        $(CC) -o foo $(objects) $(libs)
```

二、语法

条件表达式的语法为：

```
<conditional-directive>
<text-if-true>
endif
```

以及：

```
<conditional-directive>
<text-if-true>
else
<text-if-false>
endif
```

其中<conditional-directive>表示条件关键字，如“ifeq”。这个关键字有四个。第一个是我们前面所见过的“ifeq”

```
ifeq (<arg1>, <arg2>)
ifeq '<arg1>' '<arg2>'
ifeq "<arg1>" "<arg2>"
ifeq "<arg1>" '<arg2>'
ifeq '<arg1>' "<arg2>"
```

比较参数“arg1”和“arg2”的值是否相同。当然，参数中我们还可以使用 make 的函数。如：

```
ifeq ($(strip $(foo)),)
<text-if-empty>
```

endif

这个示例中使用了“strip”函数，如果这个函数的返回值是空（Empty），那么<text-if-empty>就生效。

第二个条件关键字是“ifneq”。语法是：

ifneq (<arg1>, <arg2>)

ifneq '<arg1>' '<arg2>'

ifneq "<arg1>" "<arg2>"

ifneq "<arg1>" '<arg2>'

ifneq '<arg1>' "<arg2>"

其比较参数“arg1”和“arg2”的值是否相同，如果不同，则为真。和“ifeq”类似。

第三个条件关键字是“ifdef”。语法是：

ifdef <variable-name>

如果变量<variable-name>的值非空，那到表达式为真。否则，表达式为假。当然，<variable-name>同样可以是一个函数的返回值。注意，ifdef 只是测试一个变量是否有值，其并不会把变量扩展到当前位置。还是来看两个例子：

示例一：

```
bar =
foo = $(bar)
ifdef foo
frobozz = yes
else
frobozz = no
endif
```

示例二：

```
foo =
ifdef foo
frobozz = yes
else
frobozz = no
endif
```

第一个例子中，“$(frobozz)”值是“yes”，第二个则是“no”。

第四个条件关键字是“ifndef”。其语法是：

ifndef <variable-name>

这个我就不多说了，和“ifdef”是相反的意思。

在<conditional-directive>这一行上，多余的空格是被允许的，但是不能以[Tab]键做为

开始（不然就被认为是命令）。而注释符“#”同样也是安全的。“else”和“endif”也一样，只要不是以[Tab]键开始就行了。特别注意的是，make 是在读取 Makefile 时就计算条件表达式的值，并根据条件表达式的值来选择语句，所以，你最好不要把自动化变量（如“$@”等）放入条件表达式中，因为自动化变量是在运行时才有的。而且，为了避免混乱，make 不允许把整个条件语句分成两部分放在不同的文件中。

使用函数

在 Makefile 中可以使用函数来处理变量，从而让我们的命令或是规则更为的灵活和具有智能。make 所支持的函数也不算很多，不过已经足够我们的操作了。函数调用后，函数的返回值可以当做变量来使用。

一、函数的调用语法

函数调用，很像变量的使用，也是以“$”来标识的，其语法如下：

$(<function> <arguments>)

或是

${<function> <arguments>}

这里，<function>就是函数名，make 支持的函数不多。<arguments>是函数的参数，参数间以逗号“,”分隔，而函数名和参数之间以“空格”分隔。函数调用以“$”开头，以圆括号或花括号把函数名和参数括起。感觉很像一个变量，是不是？函数中的参数可以使用变量，为了风格的统一，函数和变量的括号最好一样，如使用“$(subst a,b,$(x))”这样的形式，而不是“$(subst a,b,${x})”的形式。因为统一会更清楚，也会减少一些不必要的麻烦。

还是来看一个示例：

```
comma:= ,
empty:=
space:= $(empty) $(empty)
foo:= a b c
bar:= $(subst $(space),$(comma),$(foo))
```

在这个示例中，$(comma)的值是一个逗号。$(space)使用了$(empty)定义了一个空格，$(foo)的值是“a b c”，$(bar)的定义用，调用了函数“subst”，这是一个替换函数，这个函数有三个参数，第一个参数是被替换字串，第二个参数是替换字串，第三个参数是替换操作作用的字串。这个函数也就是把$(foo)中的空格替换成逗号，所以$(bar)的值是“a,b,c”。

二、字符串处理函数

$(subst <from>,<to>,<text>)

名称：字符串替换函数——subst。

功能：把字串<text>中的<from>字符串替换成<to>。

返回：函数返回被替换过后的字符串。

示例：

```
$(subst ee,EE,feet on the street),
```

把“feet on the street”中的“ee”替换成“EE”，返回结果是“fEEt on the strEEt”。

$(patsubst <pattern>,<replacement>,<text>)

名称：模式字符串替换函数——patsubst。

功能：查找<text>中的单词（单词以“空格”、“Tab”或“回车”“换行”分隔）是否符合模式<pattern>，如果匹配的话，则以<replacement>替换。这里，<pattern>可以包括通配符“%”，表示任意长度的字串。如果<replacement>中也包含“%”，那么，<replacement>中的这个“%”将是<pattern>中的那个“%”所代表的字串。（可以用“\”来转义，以“\%”来表示真实含义的“%”字符）

返回：函数返回被替换过后的字符串。

示例：

$(patsubst %.c,%.o,x.c.c bar.c)

把字串“x.c.c bar.c”符合模式[%.c]的单词替换成[%.o]，返回结果是“x.c.o bar.o”

备注：

这和我们前面“变量章节”说过的相关知识有点相似。如：

“$(var:<pattern>=<replacement>)”

相当于

“$(patsubst <pattern>,<replacement>,$(var))”，

而“$(var: <suffix>=<replacement>)”

则相当于

“$(patsubst %<suffix>,%<replacement>,$(var))”。

例如有：objects = foo.o bar.o baz.o，

那么，“$(objects:.o=.c)”和“$(patsubst %.o,%.c,$(objects))”是一样的。

$(strip <string>)

名称：去空格函数——strip。

功能：去掉<string>字串中开头和结尾的空字符。

返回：返回被去掉空格的字符串值。

示例：

$(strip a b c)

把字串“a b c ”去到开头和结尾的空格，结果是“a b c”。

$(findstring <find>,<in>)

名称：查找字符串函数——findstring。

功能：在字串<in>中查找<find>字串。

返回：如果找到，那么返回<find>，否则返回空字符串。

示例：

$(findstring a,a b c)

$(findstring a,b c)

第一个函数返回“a”字符串，第二个返回“”字符串（空字符串）

$(filter <pattern...>,<text>)

名称：过滤函数——filter。

功能：以<pattern>模式过滤<text>字符串中的单词，保留符合模式<pattern>的单词。可以有多个模式。

返回：返回符合模式<pattern>的字串。

示例：

```
sources := foo.c bar.c baz.s ugh.h
foo: $(sources)
        cc $(filter %.c %.s,$(sources)) -o foo
```

$(filter %.c %.s,$(sources))返回的值是“foo.c bar.c baz.s”。

$(filter-out <pattern...>,<text>)

名称：反过滤函数——filter-out。

功能：以<pattern>模式过滤<text>字符串中的单词，去除符合模式<pattern>的单词。可以有多个模式。

返回：返回不符合模式<pattern>的字串。

示例：

```
objects=main1.o foo.o main2.o bar.o
mains=main1.o main2.o
```

$(filter-out $(mains),$(objects)) 返回值是“foo.o bar.o”。

$(sort <list>)

名称：排序函数——sort。

功能：给字符串<list>中的单词排序（升序）。

返回：返回排序后的字符串。

示例：$(sort foo bar lose)返回“bar foo lose” 。

备注：sort 函数会去掉<list>中相同的单词。

$(word <n>,<text>)

名称：取单词函数——word。

功能：取字符串<text>中第<n>个单词。（从一开始）

返回：返回字符串<text>中第<n>个单词。如果<n>比<text>中的单词数要大，那么返回空字符串。

示例：$(word 2, foo bar baz)返回值是“bar”。

$(wordlist <s>,<e>,<text>)

名称：取单词串函数——wordlist。

功能：从字符串<text>中取从<s>开始到<e>的单词串。<s>和<e>是一个数字。

返回：返回字符串<text>中从<s>到<e>的单词字串。如果<s>比<text>中的单词数要大，那么返回空字符串。如果<e>大于<text>的单词数，那么返回从<s>开始，到<text>结束的单词串。

示例： $(wordlist 2, 3, foo bar baz)返回值是“bar baz”。

$(words <text>)

名称：单词个数统计函数——words。

功能：统计<text>中字符串中的单词个数。

返回：返回<text>中的单词数。

示例：$(words, foo bar baz)返回值是“3”。

备注：如果我们要取<text>中最后的一个单词，我们可以这样：$(word $(words <text>),<text>)。

$(firstword <text>)

名称：首单词函数——firstword。

功能：取字符串<text>中的第一个单词。

返回：返回字符串<text>的第一个单词。

示例：$(firstword foo bar)返回值是“foo”。

备注：这个函数可以用 word 函数来实现：$(word 1,<text>)。

以上，是所有的字符串操作函数，如果搭配混合使用，可以完成比较复杂的功能。这里，举一个现实中应用的例子。我们知道，make 使用“VPATH”变量来指定“依赖文件”的搜索路径。于是，我们可以利用这个搜索路径来指定编译器对头文件的搜索路径参数 CFLAGS，如：

override CFLAGS += $(patsubst %,-I%,$(subst :, ,$(VPATH)))

如果我们的“$(VPATH)”值是“src:../headers”，那么“$(patsubst %,-I%,$(subst :, ,$(VPATH)))”将返回“-Isrc -I../headers”，这正是 cc 或 gcc 搜索头文件路径的参数。

三、文件名操作函数

下面我们要介绍的函数主要是处理文件名的。每个函数的参数字符串都会被当做一个或是

一系列的文件名来对待。

$(dir <names...>)

名称：取目录函数——dir。

功能：从文件名序列<names>中取出目录部分。目录部分是指最后一个反斜杠（“/”）之前的部分。如果没有反斜杠，那么返回“./”。

返回：返回文件名序列<names>的目录部分。

示例： $(dir src/foo.c hacks)返回值是“src/ ./”。

$(notdir <names...>)

名称：取文件函数——notdir。

功能：从文件名序列<names>中取出非目录部分。非目录部分是指最后一个反斜杠（“/”）之后的部分。

返回：返回文件名序列<names>的非目录部分。

示例： $(notdir src/foo.c hacks)返回值是“foo.c hacks”。

$(suffix <names...>)

名称：取后缀函数——suffix。

功能：从文件名序列<names>中取出各个文件名的后缀。

返回：返回文件名序列<names>的后缀序列，如果文件没有后缀，则返回空字串。

示例：$(suffix src/foo.c src-1.0/bar.c hacks)返回值是“.c .c”。

$(basename <names...>)

名称：取前缀函数——basename。

功能：从文件名序列<names>中取出各个文件名的前缀部分。

返回：返回文件名序列<names>的前缀序列，如果文件没有前缀，则返回空字串。

示例：$(basename src/foo.c src-1.0/bar.c hacks)返回值是“src/foo src-1.0/bar hacks”。

$(addsuffix <suffix>,<names...>)

名称：加后缀函数——addsuffix。

功能：把后缀<suffix>加到<names>中的每个单词后面。

返回：返回加过后缀的文件名序列。

示例：$(addsuffix .c,foo bar)返回值是“foo.c bar.c”。

$(addprefix <prefix>,<names...>)

名称：加前缀函数——addprefix。

功能：把前缀<prefix>加到<names>中的每个单词后面。

返回：返回加过前缀的文件名序列。

示例：$(addprefix src/,foo bar)返回值是“src/foo src/bar”。

$(join <list1>,<list2>)

名称：连接函数——join。

功能：把<list2>中的单词对应地加到<list1>的单词后面。如果<list1>的单词个数要比<list2>的多，那么，<list1>中的多出来的单词将保持原样。如果<list2>的单词个数要比<list1>多，那么，<list2>多出来的单词将被复制到<list2>中。

返回：返回连接过后的字符串。

示例：$(join aaa bbb , 111 222 333)返回值是“aaa111 bbb222 333”。

四、foreach 函数

foreach 函数和别的函数非常的不一样。因为这个函数是用来做循环用的，Makefile 中的 foreach 函数几乎是仿照于 Unix 标准 Shell(/bin/sh)中的 for 语句，或是 C-Shell(/bin/csh)中的 foreach 语句而构建的。它的语法是：

$(foreach <var>,<list>,<text>)

这个函数的意思是，把参数<list>中的单词逐一取出放到参数<var>所指定的变量中，然后再执行<text>所包含的表达式。每一次<text>会返回一个字符串，循环过程中，<text>的所返回的每个字符串会以空格分隔，最后当整个循环结束时，<text>所返回的每个字符串所组成的整个字符串(以空格分隔)将会是 foreach 函数的返回值。所以，<var>最好是一个变量名，<list>可以是一个表达式，而<text>中一般会使用<var>这个参数来依次枚举<list>中的单词。举个例子：

```
names := a b c d
files := $(foreach n,$(names),$(n).o)
```

上面的例子中，$(name)中的单词会被挨个取出，并存到变量“n”中，“$(n).o”每次根据“$(n)”计算出一个值，这些值以空格分隔，最后作为 foreach 函数的返回，所以，$(files)的值是“a.o b.o c.o d.o”。

注意，foreach 中的<var>参数是一个临时的局部变量，foreach 函数执行完后，参数<var>的变量将不在作用，其作用域只在 foreach 函数当中。

五、if 函数

if 函数很像 GNU 的 make 所支持的条件语句——ifeq（参见前面所述的章节），if 函数的语法是：

$(if <condition>,<then-part>)

或是

$(if <condition>,<then-part>,<else-part>)

可见，if 函数可以包含“else”部分，或是不含。即 if 函数的参数可以是两个，也可以

是三个。<condition>参数是 if 的表达式，如果其返回的为非空字符串，那么这个表达式就相当于返回真，于是，<then-part>会被计算，否则<else-part>会被计算。而 if 函数的返回值是，如果<condition>为真（非空字符串），那个<then-part>会是整个函数的返回值，如果<condition>为假（空字符串），那么<else-part>会是整个函数的返回值，此时如果<else-part>没有被定义，那么，整个函数返回空字串。所以，<then-part>和<else-part>只会有一个被计算。

六、call 函数

call 函数是唯一一个可以用来创建新的参数化的函数。你可以写一个非常复杂的表达式，这个表达式中，你可以定义许多参数，然后你可以用 call 函数来向这个表达式传递参数。其语法是：

```
$(call <expression>,<parm1>,<parm2>,<parm3>...)
```

当 make 执行这个函数时，<expression>参数中的变量，如$(1)，$(2)，$(3)等，会被参数<parm1>，<parm2>，<parm3>依次取代。而<expression>的返回值就是 call 函数的返回值。例如：

```
reverse =  $(1) $(2)
foo = $(call reverse,a,b)
```

那么，foo 的值就是“a b”。当然，参数的次序是可以自定义的，不一定是顺序的，如：

```
reverse =  $(2) $(1)
foo = $(call reverse,a,b)
```

此时的 foo 的值就是“b a”。

七、origin 函数

origin 函数不像其它的函数，他并不操作变量的值，他只是告诉你你的这个变量是哪里来的？其语法是：

```
$(origin <variable>)
```

注意，<variable>是变量的名字，不应该是引用。所以你最好不要在<variable>中使用“$”字符。Origin 函数会以其返回值来告诉你这个变量的“出生情况”，下面，是 origin 函数的返回值:

“undefined”

如果<variable>从来没有定义过，origin 函数返回这个值“undefined”。

“default”

如果<variable>是一个默认的定义，比如“CC”这个变量，这种变量我们将在后面讲述。

“environment”

如果<variable>是一个环境变量，并且当 Makefile 被执行时，“-e”参数没有被打开。

“file”

如果<variable>这个变量被定义在 Makefile 中。

“command line”

如果<variable>这个变量是被命令行定义的。

“override”

如果<variable>是被 override 指示符重新定义的。

“automatic”

如果<variable>是一个命令运行中的自动化变量。关于自动化变量将在后面讲述。

这些信息对于我们编写 Makefile 是非常有用的，例如，假设我们有一个 Makefile 其包了一个定义文件 Make.def，在 Make.def 中定义了一个变量“bletch”，而我们的环境中也有一个环境变量“bletch”，此时，我们想判断一下，如果变量来源于环境，那么我们就把之重定义了，如果来源于 Make.def 或是命令行等非环境的，那么我们就不重新定义它。于是，在我们的 Makefile 中，我们可以这样写：

```
ifdef bletch
ifeq "$(origin bletch)" "environment"
bletch = barf, gag, etc.
endif
endif
```

当然，你也许会说，使用 override 关键字不就可以重新定义环境中的变量了吗？为什么需要使用这样的步骤？是的，我们用 override 是可以达到这样的效果，可是 override 过于粗暴，它同时会把从命令行定义的变量也覆盖了，而我们只想重新定义环境传来的，而不想重新定义命令行传来的。

八、shell 函数

shell 函数也不像其它的函数。顾名思义，它的参数应该就是操作系统 Shell 的命令。它和反引号“`”是相同的功能。这就是说，shell 函数把执行操作系统命令后的输出作为函数返回。于是，我们可以用操作系统命令以及字符串处理命令 awk，sed 等等命令来生成一个变量，如：

```
contents := $(shell cat foo)
files := $(shell echo *.c)
```

注意，这个函数会新生成一个 Shell 程序来执行命令，所以你要注意其运行性能，如果你的 Makefile 中有一些比较复杂的规则，并大量使用了这个函数，那么对于你的系统性能是有害的。特别是 Makefile 的隐晦的规则可能会让你的 shell 函数执行的次数比你想像的多得多。

九、控制 make 的函数

make 提供了一些函数来控制 make 的运行。通常，你需要检测一些运行 Makefile 时的运行时信息，并且根据这些信息来决定，你是让 make 继续执行，还是停止。

$(error <text ...>)

产生一个致命的错误，<text ...>是错误信息。注意，error 函数不会在一被使用就会产生错误信息，所以如果你把其定义在某个变量中，并在后续的脚本中使用这个变量，那么也是

可以的。例如：

示例一：

```
ifdef ERROR_001
$(error error is $(ERROR_001))
endif
```

示例二：

```
ERR = $(error found an error!)

.PHONY: err
err: ; $(ERR)
```

示例一会在变量 ERROR_001 定义了后执行时产生 error 调用，而示例二则在目录 err 被执行时才发生 error 调用。

$(warning <text ...>)

这个函数很像 error 函数，只是它并不会让 make 退出，只是输出一段警告信息，而 make 继续执行。

make 的运行

一般来说，最简单的就是直接在命令行下输入 make 命令，make 命令会找当前目录的 Makefile 来执行，一切都是自动的。但也有时你也许只想让 make 重编译某些文件，而不是整个工程，而又有的时候你有几套编译规则，你想在不同的时候使用不同的编译规则，等等。本章节就是讲述如何使用 make 命令的。

一、make 的退出码

make 命令执行后有三个退出码：

0 —— 表示成功执行。

1 —— 如果 make 运行时出现任何错误，其返回 1。

2 —— 如果你使用了 make 的“-q”选项，并且 make 使得一些目标不需要更新，那么返回 2。

Make 的相关参数我们会在后续章节中讲述。

二、指定 Makefile

前面我们说过，GNU make 找寻默认的 Makefile 的规则是在当前目录下依次找三个文件——“GNUmakefile”、“makefile”和“Makefile”。其按顺序找这三个文件，一旦找到，就开始读取这个文件并执行。当前，我们也可以给 make 命令指定一个特殊名字的 Makefile。要达到这个功能，我们要使用 make 的“-f”或是“--file”参数（“--makefile”参数也行）。例如，我们有个 Makefile 的名字是“hchen.mk”，那么，我们可以这样来让 make 来执行这个文件：

```
make  -f hchen.mk
```

如果在 make 的命令行是，你不只一次地使用了“-f”参数，那么，所有指定的 Makefile 将会被连在一起传递给 make 执行。

三、指定目标

一般来说，make 的最终目标是 Makefile 中的第一个目标，而其它目标一般是由这个目标连带出来的。这是 make 的默认行为。当然，一般来说，你的 Makefile 中的第一个目标是由许多个目标组成，你可以指示 make，让其完成你所指定的目标。要达到这一目的很简单，需在 make 命令后直接跟目标的名字就可以完成（如前面提到的“make clean”形式）任何在 Makefile 中的目标都可以被指定成终极目标，但是除了以“-”打头，或是包含了“=”的目标，因为有这些字符的目标，会被解析成命令行参数或是变量。甚至没有被我们明确写出来的目标也可以成为 make 的终极目标，也就是说，只要 make 可以找到其隐含规则推导规则，那么这个隐含目标同样可以被指定成终极目标。

有一个 make 的环境变量叫“MAKECMDGOALS”，这个变量中会存放你所指定的终极目标的列表，如果在命令行上，你没有指定目标，那么，这个变量是空值。这个变量可以让你使用在一些比较特殊的情形下。比如下面的例子：

```
sources = foo.c bar.c
ifneq ( $(MAKECMDGOALS),clean)
include $(sources:.c=.d)
endif
```

基于上面的这个例子，只要我们输入的命令不是“make clean”，那么 Makefile 会自动包含“foo.d”和“bar.d”这两个 Makefile。使用指定终极目标的方法可以很方便地让我们编译我们的程序，例如下面这个例子：

```
.PHONY: all
all: prog1 prog2 prog3 prog4
```

从这个例子中，我们可以看到，这个 Makefile 中有四个需要编译的程序——“prog1”，“prog2”，“prog3”和 “prog4”，我们可以使用“make all”命令来编译所有的目标（如果把 all 置成第一个目标，那么只需执行“make”），我们也可以使用“make prog2”来单独编译目标“prog2”。即然 make 可以指定所有 Makefile 中的目标，那么也包括“伪目标”，于是我们可以根据这种性质来让我们的 Makefile 根据指定的不同的目标来完成不同的事。在 Unix 世界中，软件发布时，特别是 GNU 这种开源软件的发布时，其 Makefile 都包含了编译、安装、打包等功能。我们可以参照这种规则来书写我们的 Makefile 中的目标。

“all”

这个伪目标是所有目标的目标，其功能一般是编译所有的目标。

“clean”

这个伪目标功能是删除所有被 make 创建的文件。

“install”

这个伪目标功能是安装已编译好的程序，其实就是把目标执行文件拷贝到指定的目标中去。

“print”

这个伪目标的功能是例出改变过的源文件。

“tar”

这个伪目标功能是把源程序打包备份。也就是一个 tar 文件。

“dist”

这个伪目标功能是创建一个压缩文件，一般是把 tar 文件压成 Z 文件。或是 gz 文件。

“TAGS”

这个伪目标功能是更新所有的目标，以备完整地重编译使用。

“check”和“test”

这两个伪目标一般用来测试 Makefile 的流程。

当然一个项目的 Makefile 中也不一定要书写这样的目标，这些东西都是 GNU 的东西，但是我想，GNU 搞出这些东西一定有其可取之处（等你的 UNIX 下的程序文件一多时你就会发现这些功能很有用了），这里只不过是说明了，如果你要书写这种功能，最好使用这种名字命名你的目标，这样规范一些，规范的好处就是——不用解释，大家都明白。而且如果你的 Makefile 中有这些功能，一是很实用，二是可以显得你的 Makefile 很专业（不是那种初学者的作品）。

四、检查规则

有时候，我们不想让我们的 Makefile 中的规则执行起来，我们只想检查一下我们的命令，或是执行的序列。于是我们可以使用 make 命令的下述参数：

“-n”

“--just-print”

“--dry-run”

“--recon”

不执行参数，这些参数只是打印命令，不管目标是否更新，把规则和连带规则下的命令打印出来，但不执行，这些参数对于我们调试 Makefile 很有用处。

“-t”

“--touch”

这个参数的意思就是把目标文件的时间更新，但不更改目标文件。也就是说，make 假装编译目标，但不是真正的编译目标，只是把目标变成已编译过的状态。

“-q”

“--question”

这个参数的行为是找目标的意思，也就是说，如果目标存在，那么其什么也不会输出，当然也不会执行编译，如果目标不存在，其会打印出一条出错信息。

“-W <file>”

“--what-if=<file>”

“--assume-new=<file>”

“--new-file=<file>”

这个参数需要指定一个文件。一般是是源文件（或依赖文件），Make 会根据规则推导来运

行依赖于这个文件的命令，一般来说，可以和“-n”参数一同使用，来查看这个依赖文件所发生的规则命令。

另外一个很有意思的用法是结合“-p”和“-v”来输出 Makefile 被执行时的信息（这个将在后面讲述）。

五、make 的参数

下面列举了所有 GNU make 3.80 版的参数定义。其它版本和产商的 make 大同小异，不过其它产商的 make 的具体参数还是请参考各自的产品文档。

“-b”

“-m”

这两个参数的作用是忽略和其它版本 make 的兼容性。

“-B”

“--always-make”

认为所有的目标都需要更新（重编译）。

“-C <dir>”

“--directory=<dir>”

指定读取 Makefile 的目录。如果有多个“-C”参数，make 的解释是后面的路径以前面的作为相对路径，并以最后的目录作为被指定目录。如：“make -C ~hchen/test -C prog”等价于“make -C ~hchen/test/prog”。

“—debug[=<options>]”

输出 make 的调试信息。它有几种不同的级别可供选择，如果没有参数，那就是输出最简单的调试信息。下面是<options>的取值：

a —— 也就是 all，输出所有的调试信息。（会非常的多）

b —— 也就是 basic，只输出简单的调试信息。即输出不需要重编译的目标。

v —— 也就是 verbose，在 b 选项的级别之上。输出的信息包括哪个 Makefile 被解析，不需要被重编译的依赖文件（或是依赖目标）等。

i —— 也就是 implicit，输出所以的隐含规则。

j —— 也就是 jobs，输出执行规则中命令的详细信息，如命令的 PID、返回码等。

m —— 也就是 Makefile，输出 make 读取 Makefile，更新 Makefile，执行 Makefile 的信息。

“-d”

相当于“--debug=a”。

“-e”
“--environment-overrides”
指明环境变量的值覆盖 Makefile 中定义的变量的值。

“-f=<file>”
“--file=<file>”
“--Makefile=<file>”
指定需要执行的 Makefile。

“-h”
“--help”
显示帮助信息。

“-i”
“--ignore-errors”
在执行时忽略所有的错误。

“-I <dir>”
“--include-dir=<dir>”
指定一个被包含 Makefile 的搜索目标。可以使用多个“-I”参数来指定多个目录。

“-j [<jobsnum>]”
“--jobs[=<jobsnum>]”
指同时运行命令的个数。如果没有这个参数，make 运行命令时能运行多少就运行多少。如果有一个以上的“-j”参数，那么仅最后一个“-j”才是有效的。(注意这个参数在 MS-DOS 中是无用的)

“-k”
“--keep-going”
出错也不停止运行。如果生成一个目标失败了，那么依赖于其上的目标就不会被执行了。

“-l <load>”
“--load-average[=<load]”
“—max-load[=<load>]”
指定 make 运行命令的负载。

“-n”
“--just-print”
“--dry-run”
“--recon”
仅输出执行过程中的命令序列，但并不执行。

“-o <file>”
“--old-file=<file>”
“--assume-old=<file>”
不重新生成的指定的<file>，即使这个目标的依赖文件新于它。

“-p”
“--print-data-base”
输出 Makefile 中的所有数据，包括所有的规则和变量。这个参数会让一个简单的 Makefile 都会输出一堆信息。如果你只是想输出信息而不想执行 Makefile，你可以使用“make -qp”命令。如果你想查看执行 Makefile 前的预设变量和规则，你可以使用“make －p －f /dev/null”。这个参数输出的信息会包含着你的 Makefile 文件的文件名和行号，所以，用这个参数来调试你的 Makefile 会是很有用的，特别是当你的环境变量很复杂的时候。

“-q”
“--question”
不运行命令，也不输出。仅仅是检查所指定的目标是否需要更新。如果是 0 则说明要更新，如果是 2 则说明有错误发生。

“-r”
“--no-builtin-rules”
禁止 make 使用任何隐含规则。

“-R”
“--no-builtin-variabes”
禁止 make 使用任何作用于变量上的隐含规则。

“-s”
“--silent”
“--quiet”
在命令运行时不输出命令的输出。

“-S”

“--no-keep-going”

“--stop”

取消“-k”选项的作用。因为有些时候，make 的选项是从环境变量“MAKEFLAGS”中继承下来的。所以你可以在命令行中使用这个参数来让环境变量中的“-k”选项失效。

“-t”

“--touch”

相当于 UNIX 的 touch 命令，只是把目标的修改日期变成最新的，也就是阻止生成目标的命令运行。

“-v”

“--version”

输出 make 程序的版本、版权等关于 make 的信息。

“-w”

“--print-directory”

输出运行 Makefile 之前和之后的信息。这个参数对于跟踪嵌套式调用 make 时很有用。

“--no-print-directory”

禁止“-w”选项。

“-W <file>”

“--what-if=<file>”

“--new-file=<file>”

“--assume-file=<file>”

假定目标<file>需要更新，如果和“-n”选项使用，那么这个参数会输出该目标更新时的运行动作。如果没有“-n”那么就像运行 UNIX 的“touch”命令一样，使得<file>的修改时间为当前时间。

“--warn-undefined-variables”

只要 make 发现有未定义的变量，那么就输出警告信息。

隐含规则

在我们使用 Makefile 时，有一些我们会经常使用，而且使用频率非常高的东西，比如，我们编译 C/C++的源程序为中间目标文件（Unix 下是[.o]文件，Windows 下是[.obj]文件）。本章

讲述的就是一些在 Makefile 中的“隐含的”，早先约定了的，不需要我们再写出来的规则。

“隐含规则”也就是一种惯例，make 会按照这种“惯例”心照不喧地来运行，那怕我们的 Makefile 中没有书写这样的规则。例如，把[.c]文件编译成[.o]文件这一规则，你根本就不用写出来，make 会自动推导出这种规则，并生成我们需要的[.o]文件。

“隐含规则”会使用一些我们系统变量，我们可以改变这些系统变量的值来定制隐含规则的运行时的参数。如系统变量“CFLAGS”可以控制编译时的编译器参数。我们还可以通过“模式规则”的方式写下自己的隐含规则。用“后缀规则”来定义隐含规则会有许多的限制。使用“模式规则”会更回得智能和清楚，但“后缀规则”可以用来保证我们 Makefile 的兼容性。我们了解了“隐含规则”，可以让其为我们更好的服务，也会让我们知道一些“约定俗成”了的东西，而不至于使得我们在运行 Makefile 时出现一些我们觉得莫名其妙的东西。当然，任何事物都是矛盾的，水能载舟，亦可覆舟，所以，有时候“隐含规则”也会给我们造成不小的麻烦。只有了解了它，我们才能更好地使用它。

一、使用隐含规则

如果要使用隐含规则生成你需要的目标，你所需要做的就是不要写出这个目标的规则。那么，make 会试图去自动推导产生这个目标的规则和命令，如果 make 可以自动推导生成这个目标的规则和命令，那么这个行为就是隐含规则的自动推导。当然，隐含规则是 make 事先约定好的一些东西。例如，我们有下面的一个 Makefile：

```
foo : foo.o bar.o
        cc  -o foo foo.o bar.o $(CFLAGS) $(LDFLAGS)
```

我们可以注意到，这个 Makefile 中并没有写下如何生成 foo.o 和 bar.o 这两目标的规则和命令。因为 make 的“隐含规则”功能会自动为我们自动去推导这两个目标的依赖目标和生成命令。make 会在自己的“隐含规则”库中寻找可以用的规则，如果找到，那么就会使用。如果找不到，那么就会报错。在上面的那个例子中，make 调用的隐含规则是，把[.o]的目标的依赖文件置成[.c]，并使用 C 的编译命令“cc -c $(CFLAGS) [.c]”来生成[.o]的目标。也就是说，我们完全没有必要写下下面的两条规则：

```
foo.o : foo.c
        cc  -c foo.c $(CFLAGS)
bar.o : bar.c
    cc  -c bar.c $(CFLAGS)
```

因为，这已经是“约定”好了的事了，make 和我们约定好了用 C 编译器“cc”生成[.o]文件的规则，这就是隐含规则。当然，如果我们为[.o]文件书写了自己的规则，那么 make 就不会自动推导并调用隐含规则，它会按照我们写好的规则忠实地执行。还有，在 make 的“隐含规则库”中，每一条隐含规则都在库中有其顺序，越靠前的则是越被经常使用的，所以，这会导致我们有些时候即使我们显示地指定了目标依赖，make 也不会管。如下面这条规则（没有命令）：

```
foo.o : foo.p
```

依赖文件“foo.p”(Pascal 程序的源文件)有可能变得没有意义。如果目录下存在了“foo.c”

文件，那么我们的隐含规则一样会生效，并会通过“foo.c”调用 C 的编译器生成 foo.o 文件。因为，在隐含规则中，Pascal 的规则出现在 C 的规则之后，所以，make 找到可以生成 foo.o 的 C 的规则就不再寻找下一条规则了。如果你确实不希望任何隐含规则推导，那么，你就不要只写出“依赖规则”，而不写命令。

二、隐含规则一览

这里我们将讲述所有预先设置（也就是 make 内建）的隐含规则，如果我们不明确地写下规则，那么，make 就会在这些规则中寻找所需要规则和命令。当然，我们也可以使用 make 的参数“-r”或“--no-builtin-rules”选项来取消所有的预设置的隐含规则。当然，即使是我们指定了“-r”参数，某些隐含规则还是会生效，因为有许多的隐含规则都是使用了“后缀规则”来定义的，所以，只要隐含规则中有“后缀列表”（也就一系统定义在目标.SUFFIXES 的依赖目标），那么隐含规则就会生效。默认的后缀列表是：.out, .a, .ln, .o, .c, .cc, .C, .p, .f, .F, .r, .y, .l, .s, .S, .mod, .sym, .def, .h, .info, .dvi, .tex, .texinfo, .texi, .txinfo, .w, .ch .web, .sh, .elc, .el。具体的细节，我们会在后面讲述。还是先来看一看常用的隐含规则吧。

1、编译 C 程序的隐含规则。

“<n>.o”的目标的依赖目标会自动推导为“<n>.c”，并且其生成命令是“$(CC) -c $(CPPFLAGS) $(CFLAGS)”

2、编译 C++程序的隐含规则。

“<n>.o”的目标的依赖目标会自动推导为“<n>.cc”或是“<n>.C”，并且其生成命令是“$(CXX) -c $(CPPFLAGS) $(CFLAGS)”。（建议使用“.cc”作为 C++源文件的后缀，而不是“.C”）

3、编译 Pascal 程序的隐含规则。

“<n>.o”的目标的依赖目标会自动推导为“<n>.p”，并且其生成命令是“$(PC) -c $(PFLAGS)”。

4、编译 Fortran/Ratfor 程序的隐含规则。

“<n>.o”的目标的依赖目标会自动推导为“<n>.r”或“<n>.F”或“<n>.f”，并且其生成命令是:

“.f” “$(FC) -c $(FFLAGS)”

“.F” “$(FC) -c $(FFLAGS) $(CPPFLAGS)”

“.f” “$(FC) -c $(FFLAGS) $(RFLAGS)”

5、预处理 Fortran/Ratfor 程序的隐含规则。

“<n>.f”的目标的依赖目标会自动推导为“<n>.r”或“<n>.F”。这个规则只是转换 Ratfor 或有预处理的 Fortran 程序到一个标准的 Fortran 程序。其使用的命令是：

“.F” “$(FC) -F $(CPPFLAGS) $(FFLAGS)”

“.r” “$(FC) -F $(FFLAGS) $(RFLAGS)”

6、编译 Modula-2 程序的隐含规则。

“<n>.sym”的目标的依赖目标会自动推导为“<n>.def”，并且其生成命令是：“$(M2C)

$(M2FLAGS) $(DEFFLAGS)”。“<n.o>” 的目标的依赖目标会自动推导为“<n>.mod”，并且其生成命令是：“$(M2C) $(M2FLAGS) $(MODFLAGS)”。

7、汇编和汇编预处理的隐含规则。

“<n>.o” 的目标的依赖目标会自动推导为“<n>.s”，默认使用编译品“as”，并且其生成命令是：“$(AS) $(ASFLAGS)”。“<n>.s” 的目标的依赖目标会自动推导为“<n>.S”，默认使用C 预编译器“cpp”，并且其生成命令是：“$(AS) $(ASFLAGS)”。

8、链接 Object 文件的隐含规则。

“<n>”目标依赖于“<n>.o”，通过运行 C 的编译器来运行链接程序生成（一般是“ld”），其生成命令是：“$(CC) $(LDFLAGS) <n>.o $(LOADLIBES) $(LDLIBS)”。这个规则对于只有一个源文件的工程有效，同时也对多个 Object 文件（由不同的源文件生成）的也有效。例如如下规则：

```
x : y.o z.o
```

并且“x.c”、“y.c”和“z.c”都存在时，隐含规则将执行如下命令：

```
cc -c x.c -o x.o
cc -c y.c -o y.o
cc -c z.c -o z.o
cc x.o y.o z.o -o x
rm -f x.o
rm -f y.o
rm -f z.o
```

如果没有一个源文件（如上例中的 x.c）和你的目标名字（如上例中的 x）相关联，那么，你最好写出自己的生成规则，不然，隐含规则会报错的。

9、Yacc C 程序时的隐含规则。

“<n>.c”的依赖文件被自动推导为“n.y”（Yacc 生成的文件），其生成命令是：“$(YACC) $(YFALGS)”。（“Yacc”是一个语法分析器，关于其细节请查看相关资料）

10、Lex C 程序时的隐含规则。

“<n>.c”的依赖文件被自动推导为“n.l”（Lex 生成的文件），其生成命令是：“$(LEX) $(LFALGS)”。（关于“Lex”的细节请查看相关资料）

11、Lex Ratfor 程序时的隐含规则。

“<n>.r”的依赖文件被自动推导为“n.l”（Lex 生成的文件），其生成命令是：“$(LEX) $(LFALGS)”。

12、从 C 程序、Yacc 文件或 Lex 文件创建 Lint 库的隐含规则。

“<n>.ln”（lint 生成的文件）的依赖文件被自动推导为“n.c”，其生成命令是：“$(LINT) $(LINTFALGS) $(CPPFLAGS) -i”。对于“<n>.y”和“<n>.l”也是同样的规则。

三、隐含规则使用的变量

在隐含规则中的命令中，基本上都是使用了一些预先设置的变量。你可以在你的 Makefile

中改变这些变量的值，或是在 make 的命令行中传入这些值，或是在你的环境变量中设置这些值，无论怎么样，只要设置了这些特定的变量，那么其就会对隐含规则起作用。当然，你也可以利用 make 的“-R”或“--no－builtin-variables”参数来取消你所定义的变量对隐含规则的作用。例如，第一条隐含规则——编译 C 程序的隐含规则的命令是“$(CC) －c $(CFLAGS) $(CPPFLAGS)”。Make 默认的编译命令是“cc”，如果你把变量“$(CC)”重定义成“gcc”，把变量“$(CFLAGS)”重定义成“-g”，那么，隐含规则中的命令全部会以“gcc －c -g $(CPPFLAGS)”的样子来执行了。我们可以把隐含规则中使用的变量分成两种：一种是命令相关的，如“CC”；一种是参数相的关，如“CFLAGS”。下面是所有隐含规则中会用到的变量：

1、关于命令的变量。

AR

函数库打包程序。默认命令是“ar”。

AS

汇编语言编译程序。默认命令是“as”。

CC

C 语言编译程序。默认命令是“cc”。

CXX

C++语言编译程序。默认命令是“g++”。

CO

从 RCS 文件中扩展文件程序。默认命令是“co”。

CPP

C 程序的预处理器（输出是标准输出设备）。默认命令是“$(CC) －E”。

FC

Fortran 和 Ratfor 的编译器和预处理程序。默认命令是“f77”。

GET

从 SCCS 文件中扩展文件的程序。默认命令是“get”。

LEX

Lex 方法分析器程序（针对于 C 或 Ratfor）。默认命令是“lex”。

PC

Pascal 语言编译程序。默认命令是“pc”。

YACC

Yacc 文法分析器（针对于 C 程序）。默认命令是“yacc”。

YACCR

Yacc 文法分析器（针对于 Ratfor 程序）。默认命令是“yacc －r”。

MAKEINFO

转换 Texinfo 源文件（.texi）到 Info 文件程序。默认命令是“makeinfo”。

TEX

从 TeX 源文件创建 TeX DVI 文件的程序。默认命令是“tex”。

TEXI2DVI

从 Texinfo 源文件创建军 TeX DVI 文件的程序。默认命令是“texi2dvi”。

WEAVE

转换 Web 到 TeX 的程序。默认命令是“weave”。

CWEAVE

转换 C Web 到 TeX 的程序。默认命令是“cweave”。

TANGLE

转换 Web 到 Pascal 语言的程序。默认命令是“tangle”。

CTANGLE

转换 C Web 到 C。默认命令是“ctangle”。

RM

删除文件命令。默认命令是“rm -f”。

2、关于命令参数的变量

下面的这些变量都是相关上面的命令的参数。如果没有指明其默认值，那么其默认值都是空。

ARFLAGS

函数库打包程序 AR 命令的参数。默认值是“rv”。

ASFLAGS

汇编语言编译器参数。(当明显地调用“.s”或“.S”文件时)。

CFLAGS

C 语言编译器参数。

CXXFLAGS

C++语言编译器参数。

COFLAGS

RCS 命令参数。

CPPFLAGS

C 预处理器参数。(C 和 Fortran 编译器也会用到)。

FFLAGS

Fortran 语言编译器参数。

GFLAGS

SCCS “get”程序参数。

LDFLAGS

链接器参数。(如:“ld”)

LFLAGS

Lex 文法分析器参数。

PFLAGS

Pascal 语言编译器参数。

RFLAGS

Ratfor 程序的 Fortran 编译器参数。

YFLAGS

Yacc 文法分析器参数。

四、隐含规则链

有些时候，一个目标可能被一系列的隐含规则所作用。例如，一个[.o]的文件生成，可能会是先被 Yacc 的[.y]文件先成[.c]，然后再被 C 的编译器生成。我们把这一系列的隐含规则叫做“隐含规则链”。在上面的例子中，如果文件[.c]存在，那么就直接调用 C 的编译器的隐含规则，如果没有[.c]文件，但有一个[.y]文件，那么 Yacc 的隐含规则会被调用，生成[.c]文件，然后，再调用 C 编译的隐含规则最终由[.c]生成[.o]文件，达到目标。我们把这种[.c]的文件（或是目标），叫做中间目标。不管怎么样，make 会努力自动推导生成目标的一切方法，不管中间目标有多少，其都会执着地把所有的隐含规则和你书写的规则全部合起来分析，努力达到目标，所以，有些时候，可能会让你觉得奇怪，怎么我的目标会这样生成？怎么我的 Makefile 发疯了？在默认情况下，对于中间目标，它和一般的目标有两个地方所不同：第一个不同是除非中间的目标不存在，才会引发中间规则。第二个不同的是，只要目标成功产生，那么，产生最终目标过程中，所产生的中间目标文件会被以“rm -f”删除。

通常，一个被 Makefile 指定成目标或是依赖目标的文件不能被当作中介。然而，你可以明显地说明一个文件或是目标是中介目标，你可以使用伪目标“.INTERMEDIATE”来强制声明。（如：.INTERMEDIATE ： mid ）你也可以阻止 make 自动删除中间目标，要做到这一点，你可以使用伪目标“.SECONDARY”来强制声明（如：.SECONDARY : sec）。你还可以把你的目标，以模式的方式来指定（如：%.o）成伪目标“.PRECIOUS”的依赖目标，以保存被隐含规则所生成的中间文件。在“隐含规则链”中，禁止同一个目标出现两次或两次以上，这样一来，就可防止在 make 自动推导时出现无限递归的情况。

Make 会优化一些特殊的隐含规则，而不生成中间文件。如，从文件“foo.c”生成目标程序“foo”，按道理，make 会编译生成中间文件“foo.o”，然后链接成“foo”，但在实际情况下，这一动作可以被一条“cc”的命令完成（cc -o foo foo.c），于是优化过的规则就不会生成中间文件。

五、定义模式规则

你可以使用模式规则来定义一个隐含规则。一个模式规则就好像一个一般的规则，只是在规则中，目标的定义需要有"%"字符。"%"的意思是表示一个或多个任意字符。在依赖目标中同样可以使用"%"，只是依赖目标中的"%"的取值，取决于其目标。有一点需要注意的是，"%"的展开发生在变量和函数的展开之后，变量和函数的展开发生在 make 载入 Makefile 时，而模式规则中的"%"则发生在运行时。

1、模式规则介绍

模式规则中，至少在规则的目标定义中要包含"%"，否则，就是一般的规则。目标中的"%"定义表示对文件名的匹配，"%"表示长度任意的非空字符串。例如："%.c"表示以".c"结尾的文件名（文件名的长度至少为 3），而"s.%.c"则表示以"s."开头，".c"结尾的文件名（文件名的长度至少为 5）。如果"%"定义在目标中，那么，目标中的"%"的值决定了依赖目标中的"%"的值，也就是说，目标中的模式的"%"决定了依赖目标中"%"的样子。例如有一个模式规则如下：

%.o : %.c ; <command .....>

其含义是，指出了怎么从所有的[.c]文件生成相应的[.o]文件的规则。如果要生成的目标是"a.o b.o"，那么"%c"就是"a.c b.c"。一旦依赖目标中的"%"模式被确定，那么，make 会被要求去匹配当前目录下所有的文件名，一旦找到，make 就会规则下的命令，所以，在模式规则中，目标可能会是多个的，如果有模式匹配出多个目标，make 就会产生所有的模式目标，此时，make 关心的是依赖的文件名和生成目标的命令这两件事。

2、模式规则示例

下面这个例子表示了,把所有的[.c]文件都编译成[.o]文件.

```
%.o : %.c
        $(CC) -c $(CFLAGS) $(CPPFLAGS) $< -o $@
```

其中，"$@"表示所有的目标的挨个值，"$<"表示了所有依赖目标的挨个值。这些奇怪的变量我们叫"自动化变量"，后面会详细讲述。

下面的这个例子中有两个目标是模式的：

```
%.tab.c %.tab.h: %.y
        bison -d $<
```

这条规则告诉 make 把所有的[.y]文件都以"bison -d <n>.y"执行，然后生成"<n>.tab.c"和"<n>.tab.h"文件。(其中，"<n>"表示一个任意字符串)。如果我们的执行程序"foo"依赖于文件"parse.tab.o"和"scan.o"，并且文件"scan.o"依赖于文件"parse.tab.h"，如果"parse.y"文件被更新了,那么根据上述的规则,"bison -d parse.y"就会被执行一次,于是,"parse.tab.o"和"scan.o"的依赖文件就齐了。(假设，"parse.tab.o"由"parse.tab.c"生成，和"scan.o"由"scan.c"生成，而"foo"由"parse.tab.o"和"scan.o"链接生成，而且 foo 和其[.o]文件的依赖关系也写好，那么，所有的目标都会得到满足)

3、自动化变量

在上述的模式规则中，目标和依赖文件都是一系例的文件，那么我们如何书写一个命令来完成从不同的依赖文件生成相应的目标？因为在每一次的对模式规则的解析时，都会是不同的目标和依赖文件。自动化变量就是完成这个功能的。在前面，我们已经对自动化变量有所提涉，相信你看到这里已对它有一个感性认识了。所谓自动化变量，就是这种变量会把模式中所定义的一系列的文件自动地挨个取出，直至所有的符合模式的文件都取完了。这种自动化变量只应出现在规则的命令中。下面是所有的自动化变量及其说明：

$@

表示规则中的目标文件集。在模式规则中，如果有多个目标，那么，"$@"就是匹配于目标中模式定义的集合。

$%

仅当目标是函数库文件中，表示规则中的目标成员名。例如，如果一个目标是"foo.a(bar.o)"，那么，"$%"就是"bar.o"，"$@"就是"foo.a"。如果目标不是函数库文件（Unix 下是[.a]，Windows 下是[.lib]），那么，其值为空。

$<

依赖目标中的第一个目标名字。如果依赖目标是以模式（即"%"）定义的，那么"$<"将是符合模式的一系列的文件集。注意，其是一个一个取出来的。

$?

所有比目标新的依赖目标的集合。以空格分隔。

$^

所有的依赖目标的集合。以空格分隔。如果在依赖目标中有多个重复的，那个这个变量会去除重复的依赖目标，只保留一份。

$+

这个变量很像"$^"，也是所有依赖目标的集合。只是它不去除重复的依赖目标。

$*

这个变量表示目标模式中"%"及其之前的部分。如果目标是"dir/a.foo.b"，并且目标的模式是"a.%.b"，那么，"$*"的值就是"dir/a.foo"。这个变量对于构造有关联的文件名是比较有较。如果目标中没有模式的定义，那么"$*"也就不能被推导出，但是，如果目标文件的后缀是 make 所识别的，那么"$*"就是除了后缀的那一部分。例如：如果目标是"foo.c"，因为".c"是 make 所能识别的后缀名，所以，"$*"的值就是"foo"。这个特性是 GNU make 的，很有可能不兼容于其它版本的 make，所以，你应该尽量避免使用"$*"，除非是在隐含规则或是静态模式中。如果目标中的后缀是 make 所不能识别的，那么"$*"就是空值。

当你希望只对更新过的依赖文件进行操作时，"$?"在显式规则中很有用，例如，假设有一个函数库文件叫"lib"，其由其它几个 object 文件更新。那么把 object 文件打包的比较有效率的 Makefile 规则是：

```
lib : foo.o bar.o lose.o win.o
        ar r lib $?
```

在上述所列出来的自动量变量中。四个变量（$@、$<、$%、$*）在扩展时只会有一个文件，而另三个的值是一个文件列表。这七个自动化变量还可以取得文件的目录名或是在当前目录下的符合模式的文件名，只需要搭配上"D"或"F"字样。这是 GNU make 中老版本的特性，在新版本中，我们使用函数"dir"或"notdir"就可以做到了。"D"的含义就是 Directory，就是目录，"F"的含义就是 File，就是文件。

下面是对于上面的七个变量分别加上"D"或是"F"的含义：

$(@D)

表示"$@"的目录部分（不以斜杠作为结尾），如果"$@"值是"dir/foo.o"，那么"$(@D)"就是"dir"，而如果"$@"中没有包含斜杠的话，其值就是"."（当前目录）。

$(@F)

表示"$@"的文件部分，如果"$@"值是"dir/foo.o"，那么"$(@F)"就是"foo.o"，"$(@F)"相当于函数"$(notdir $@)"。

"$(*D)"

"$(*F)"

和上面所述的同理，也是取文件的目录部分和文件部分。对于上面的那个例子，"$(*D)"返回"dir"，而"$(*F)"返回"foo"

"$(%D)"

"$(%F)"

分别表示了函数包文件成员的目录部分和文件部分。这对于形同"archive(member)"形式的目标中的"member"中包含了不同的目录很有用。

"$(<D)"

"$(<F)"

分别表示依赖文件的目录部分和文件部分。

"$(^D)"

"$(^F)"

分别表示所有依赖文件的目录部分和文件部分。（无相同的）

"$(+D)"

"$(+F)"

分别表示所有依赖文件的目录部分和文件部分。（可以有相同的）

"$(?D)"

"$(?F)"

分别表示被更新的依赖文件的目录部分和文件部分。

最后想提醒一下的是，对于"$<"，为了避免产生不必要的麻烦，我们最好给$后面的那个特定字符都加上圆括号，比如，"$(<)"就要比"$<"要好一些。还得要注意的是，这些变量只使用在规则的命令中，而且一般都是"显式规则"和"静态模式规则"（参见前面"书写规则"一章）。其在隐含规则中并没有意义。

4、模式的匹配

一般来说，一个目标的模式有一个有前缀或是后缀的"%"，或是没有前后缀，直接就是一个"%"。因为"%"代表一个或多个字符，所以在定义好了的模式中，我们把"%"所匹配的内容叫做"茎"，例如"%.c"所匹配的文件"test.c"中"test"就是"茎"。因为在目标和依赖目标中同时有"%"时，依赖目标的"茎"会传给目标，当做目标中的"茎"。

当一个模式匹配包含有斜杠（实际也不经常包含）的文件时，那么在进行模式匹配时，目录部分会首先被移开，然后进行匹配，成功后，再把目录加回去。在进行"茎"的传递时，我们

需要知道这个步骤。例如有一个模式"e%t"，文件"src/eat"匹配于该模式，于是"src/a"就是其"茎"，如果这个模式定义在依赖目标中，而被依赖于这个模式的目标中又有个模式"c%r"，那么，目标就是"src/car"。("茎"被传递)

5、重载内建隐含规则

你可以重载内建的隐含规则(或是定义一个全新的)，例如你可以重新构造和内建隐含规则不同的命令，如：

```
%.o : %.c
        $(CC) -c $(CPPFLAGS) $(CFLAGS) -D$(date)
```

你可以取消内建的隐含规则，只要不在后面写命令就行。如：

```
%.o : %.s
```

同样，你也可以重新定义一个全新的隐含规则，其在隐含规则中的位置取决于你在哪里写下这个规则。朝前的位置就靠前。

六、老式风格的"后缀规则"

后缀规则是一个比较老式的定义隐含规则的方法。后缀规则会被模式规则逐步地取代。因为模式规则更强更清晰。为了和老版本的 Makefile 兼容，GNU make 同样兼容于这些东西。后缀规则有两种方式："双后缀"和"单后缀"。双后缀规则定义了一对后缀：目标文件的后缀和依赖目标（源文件）的后缀。如".c.o"相当于"%o : %c"。单后缀规则只定义一个后缀，也就是源文件的后缀。如".c"相当于"% : %.c"。

后缀规则中所定义的后缀应该是 make 所认识的，如果一个后缀是 make 所认识的，那么这个规则就是单后缀规则，而如果两个连在一起的后缀都被 make 所认识，那就是双后缀规则。例如：".c"和".o"都是 make 所知道。因而，如果你定义了一个规则是".c.o"那么其就是双后缀规则，意义就是".c"是源文件的后缀，".o"是目标文件的后缀。如下示例：

```
.c.o:
        $(CC) -c $(CFLAGS) $(CPPFLAGS) -o $@ $<
```

后缀规则不允许任何的依赖文件，如果有依赖文件的话，那就不是后缀规则，那些后缀统统被认为是文件名，如：

```
.c.o: foo.h
        $(CC) -c $(CFLAGS) $(CPPFLAGS) -o $@ $<
```

这个例子，就是说，文件".c.o"依赖于文件"foo.h"，而不是我们想要的这样：

```
%.o: %.c foo.h
        $(CC) -c $(CFLAGS) $(CPPFLAGS) -o $@ $<
```

后缀规则中，如果没有命令，那是毫无意义的。因为他也不会移去内建的隐含规则。而要让 make 知道一些特定的后缀，我们可以使用伪目标".SUFFIXES"来定义或是删除，如：

```
.SUFFIXES: .hack .win
```

把后缀.hack 和.win 加入后缀列表中的末尾。

```
.SUFFIXES:                # 删除默认的后缀
```

.SUFFIXES: .c .o .h # 定义自己的后缀

先清楚默认后缀，后定义自己的后缀列表。make 的参数"-r"或"-no-builtin-rules"也会使用得默认的后缀列表为空。而变量"SUFFIXE"被用来定义默认的后缀列表，你可以用".SUFFIXES"来改变后缀列表，但请不要改变变量"SUFFIXE"的值。

七、隐含规则搜索算法

比如我们有一个目标叫 T。下面是搜索目标 T 的规则的算法。请注意，在下面，我们没有提到后缀规则，原因是，所有的后缀规则在 Makefile 被载入内存时，会被转换成模式规则。如果目标是"archive(member)"的函数库文件模式，那么这个算法会被运行两次，第一次是找目标 T，如果没有找到的话，那么进入第二次，第二次会把"member"当作 T 来搜索。

1、把 T 的目录部分分离出来。叫 D，而剩余部分叫 N。（如：如果 T 是"src/foo.o"，那么，D 就是"src/"，N 就是"foo.o"）

2、创建所有匹配于 T 或是 N 的模式规则列表。

3、如果在模式规则列表中有匹配所有文件的模式，如"%"，那么从列表中移除其它的模式。

4、移除列表中没有命令的规则。

5、对于第一个在列表中的模式规则：

1）推导其"茎"S，S 应该是 T 或是 N 匹配于模式中"%"非空的部分。

2）计算依赖文件。把依赖文件中的"%"都替换成"茎"S。如果目标模式中没有包含斜框字符，而把 D 加在第一个依赖文件的开头。

3）测试是否所有的依赖文件都存在或是理当存在。（如果有一个文件被定义成另外一个规则的目标文件，或者是一个显式规则的依赖文件，那么这个文件就叫"理当存在"）

4）如果所有的依赖文件存在或是理当存在，或是就没有依赖文件。那么这条规则将被采用，退出该算法。

6、如果经过第 5 步，没有模式规则被找到，那么就做更进一步的搜索。对于存在于列表中的第一个模式规则：

1）如果规则是终止规则，那就忽略它，继续下一条模式规则。

2）计算依赖文件。（同第 5 步）

3）测试所有的依赖文件是否存在或是理当存在。

4）对于不存在的依赖文件，递归调用这个算法查找他是否可以被隐含规则找到。

5）如果所有的依赖文件存在或是理当存在，或是就根本没有依赖文件。那么这条规则被采用，退出该算法。

7、如果没有隐含规则可以使用，查看".DEFAULT"规则，如果有，采用，把".DEFAULT"的命令给 T 使用。

一旦规则被找到，就会执行其相当的命令，而此时，我们的自动化变量的值才会生成。