Hi3518A/Hi3518C/Hi3518E/Hi3516C Linux 开发环境

用户指南

文档版本　01

发布日期　2014-02-26

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：　深圳市龙岗区坂田华为基地华为电气生产中心　邮编：518129

网址：　http://www.hisilicon.com

客户服务电话：　+86-755-28788858

客户服务传真：　+86-755-28357515

客户服务邮箱：　support@hisilicon.com

前 言

概述

本文档介绍 Linux 开发环境。Linux 开发环境的搭建、HiBoot、Linux 内核、根文件系统以及内核和根文件系统的烧写，以及创建网络开发环境和如何启动 Linux 开发应用程序。

本文档主要提供让客户更快地了解 Linux 开发环境指导。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3518A 芯片	V100
Hi3518C 芯片	V100
Hi3518E 芯片	V100
Hi3516C 芯片	V100

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2014-02-26	01	**补充 Hi3518E 的信息。**
2012-12-26	00B30	**补充 Hi3516C 的信息。** **第 2 章 U-boot** 删除原 2.3 编译 **U-boot 章节。** **第 3 章 Lniux** 3.2 配置内核中新增编译参数 CROSS_COMPILE 及 2 种工具链的相关说明。
2012-11-25	00B20	**第 3 章 Lniux** 3.2 配置内核和 3.3 编译内核并生成内核镜像 uImage 中描述更新。
2012-08-30	00B10	第 2 次版本发布。
2012-08-15	00B01	第 1 次临时版本发布。

目 录

插图目录

表格目录

1 开发环境

说明
Hi3518A、Hi3518C、Hi3518E 及 Hi3516C Linux 开发环境的相关原理和操作基本一致。与芯片相关的操作本文后面的均以 Hi3518A 为例进行说明。

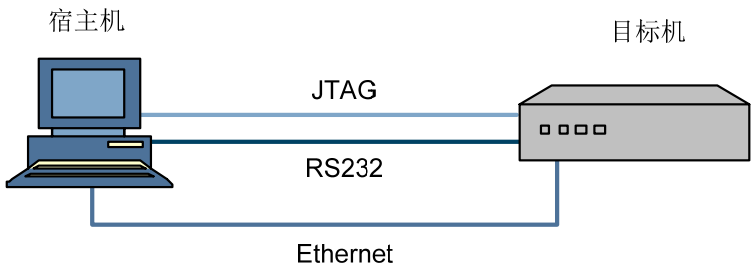
1.1 嵌入式开发环境

由于嵌入式单板的资源有限，不能在单板上运行开发和调试工具，通常需要交叉编译调试的方式进行开发和调试，即“宿主机＋目标机（评估板）”的形式。宿主机和目标机一般采用串口连接，也可同时通过网口或者 JTAG 连接，如图 1-1 所示。

宿主机和目标机的处理器一般不相同。宿主机需要建立适合于目标机的交叉编译环境。程序在宿主机上经过“编译－连接－定位”得到可执行文件。通过一定的方法将可执行文件烧写到目标机中，然后在目标机上运行。

目标机上的 Bootloader 启动后，目标机中的操作信息通过串口或者网口输出到宿主机上显示。在宿主机上的控制台中输入命令，可以控制目标机。

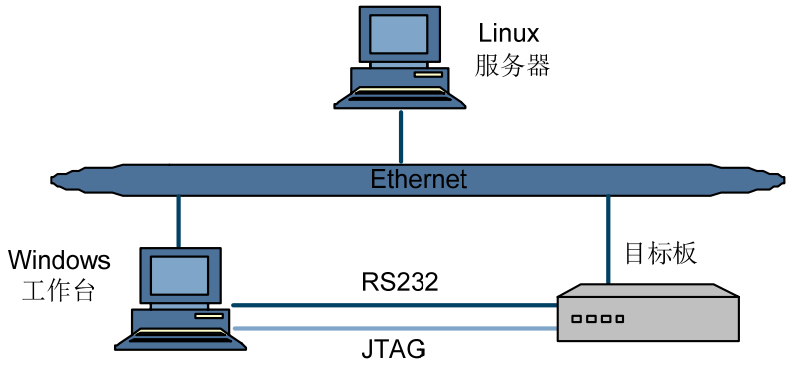
图1-1 嵌入式开发图例



1.2 Linux 开发环境

Linux 开发环境通常包括 Linux 服务器、Windows 工作台和 DMEB（目标板），三者同处于一个网络中，如图 1-2 所示。

图1-2 Linux 开发环境



在 Linux 服务器上建立交叉编译环境，Windows 工作台通过串口和网口与单板连接，开发人员可以在 Windows 工作台中进行程序开发或者远程登录到 Linux 服务器进行程序开发。各部分具体软件介绍如表 1-1 所示。

说明

开发环境中使用了 Windows 工作台，实际上很多工作也可以在 Linux 服务器上完成，如使用 minicom 代替超级终端等，用户可自行选择。

表1-1 Linux 开发环境的各部分软件描述

软件		描述
Windows 工作台	操作系统	Windows 98/me/2000/XP。
	应用软件	putty、超级终端、tftp 服务器、ADS/RealView Debugger 等软件。
Linux 服务器	操作系统	无特别要求，可为 Redhat、Debian 等。内核版本支持 2.6.18 及以上版本。安装时建议选择完全安装。
	应用软件	NFS、telnetd、samba、VIM、arm 交叉编译环境（Gcc 版本 4.4.1）等。 其他应用软件根据具体开发需要而定，通常系统都已默认安装，只要适当配置即可。
目标板	引导程序	U-boot。
	操作系统	Hisilicon Linux（简称 HiLinux）。HiLinux 内核基于 Linux 标准内核 3.0.y 版本移植开发，根文件系统基于 busybox 1.16.1 版本制作而成。

软件		描述
	应用软件	包含 telnetd、gdb server 等 Linux 常用命令。
	程序开发库	uClibc-0.9.32.1 版本； glibc-2.11.1 版本。

1.3 搭建 Linux 开发环境

1.3.1 安装 Linux 服务器

建议选择常用的 Linux 发行版，便于寻找各类技术资源。例如：

- RedHat 较新的发行版如 RedHat Fedora Core 系列和 Redhat Enterprise Linux、Red Hat 3.4.4-2。
- RedHat 较老的发行版如 RedHat 9.0 等。

推荐使用较新版本，以方便获取各类资源，如 Fedora Core 系列、SUCE10、Ubuntu9。

Debian 的各类发行版也是常用的。使用 Debian 的好处是各类安装包都可以随时在线更新，各类软件包资源也很丰富。

1.3.2 安装交叉编译工具

注意

使用从网络等渠道得到的交叉编译工具可能存在与使用的内核并不配套，造成开发过程中出现一些不可预料的问题。

可以使用从其它渠道得到的 ARM 交叉编译工具（如：网络下载），用户需要熟悉交叉编译环境的安装及使用过程，建议解压 SDK 中工具链的压缩包，然后使用 **sudo** 或 **root** 权限执行 **cross.install** 即可完成安装。

1.3.3 安装 SDK

说明

Hi3518A、Hi3518C、Hi3518E 和 Hi3516C 的 SDK 软件安装包包名都是 Hi3518_SDK_VXX.tgz（XX 是版本号）。

Hi3518A SDK 是基于 Hi3518A DMEB 的软件开发包，包含了在 Linux 相关应用开发时使用的各种工具及其源代码，是用户开发中最基本的软件平台。将 Hi3518A SDK 安装到 Linux 服务器中的步骤如下：

1. 拷贝。将 Hi3518_SDK_VXX.tgz（XX 是版本号）拷贝到 Linux 服务器上。

2. 解压。解压文件，使用命令：tar –zxf　Hi3518_SDK_VXX.tgz。
3. 如果过程中没有提示信息，请等待命令执行完毕。
4. 安装。解压完成后，进入 Hi3518_SDK_VXX 目录，执行./ sdk.unpack，执行完毕后安装成功。

   如果用户没有 root 权限，安装过程中必要的时候会提示输入 root 密码或 sudo 密码。

   ----结束

2 U-boot

2.1 U-boot 简介

U-boot 是在 U-boot-2010.06 基础上进行开发的。

Bootloader 是在操作系统内核运行之前运行的一段小程序。通过这段小程序，可以实现以下功能：

- 初始化硬件设备。
- 建立内存空间的映射图。
- 使系统的软硬件环境处于一个合适的状态，为最终调用操作系统内核准备好正确的环境。

U-boot 除了作为一个 Bootloader 外，还是一个烧写器。在 U-boot 下，可以通过串口、网口下载 Linux 内核或者应用程序到内存或 Flash 中。

说明
如无特别说明，本文中 U-boot 是包含 SPI Flash、NAND Flash 两种启动合一的版本，它们的启动、编译和烧写过程大致相同，涉及到的命令和环境变量略有不同，具体请参见“2.4 U-boot 常用命令”和“2.5 U-boot 环境变量”。

2.2 启动 U-boot

给 Hi3518A DMEB 上电后，在控制台上出现命令提示符。Hi3518A DMEB 的标准输入、标准输出重定位到 UART0。UART0 连接到调试主机（Host）上，调试主机是 Windows 工作台，采用超级终端（如果调试主机是 Linux 服务器，采用 MiniCOM）。UART0 的连接设置为：

- 波特率：115200
- 数据位：8
- 奇偶校验：无
- 停止位：1
- 流控：无

系统上电后，控制台上有如下类似的信息显示，表示 U-boot 已经启动：

```
U-Boot 2010.06 (Mar 31 2011 - 16:27:04)
```

```
DRAM:  1 GiB
NAND:  Special Nand id table Version 1.21
Nand ID: 0xAD 0xDC 0x80 0x95 0xAD 0xDC 0x80 0x95
Nand(Hardware): Block:128K Page:2K Ecc:1bit Chip:512M OOB:64Byte
512 MiB
In:    serial
Out:   serial
Err:   serial
hisilicon #
```

2.3 烧写 U-boot

编译和烧写 U-boot 的具体内容请参见《Hi3518A/Hi3518C/Hi3518E/Hi3516C U-boot 移植应用指南》。

2.4 U-boot 常用命令

U-boot 常用命令的描述如表 2-1 所示。

说明

U-boot 支持命令自动补齐，当输入命令的部分字母时，按下 Tab 键，系统将自动补齐或者列出可能的命令列表。

表2-1 U-boot 常用命令说明

命令	描述
?	得到所有命令列表或者列出某个命令的帮助。 用法：？ [command ...] 说明：列出命令的帮助信息。当不带参数时，列出所有命令及简要说明。
help	help 同？ 。
printenv	打印环境变量。 用法：printenv [name ...] 说明：打印环境变量。当不带参数时，打印所有变量。
setenv	设置或者删除变量。 用法：setenv name [value] 说明：name 一般是 U-boot 环境变量的名字，也可以是用户自定义的变量。当 value 为空时，删除变量“name”，否则设置变量“name”，且值为“value”。

命令	描述
saveenv	保存变量。 用法：saveenv 说明：保存变量及其值至 Flash。
ping	用于简单判断目的主机网络状态或本机网络工作状态。 用法：ping <ipaddr> 说明：ipaddr 表示目的主机的 IP。当网络工作正常时，结果显示“host <ipaddr> is alive”，否则显示“ping failed;host <ipaddr> is not alive”。
loadb	通过串口 Kermit 协议下载二进制文件。 用法：loadb [addr] [baud] 说明：addr 参数为存储文件的地址，baud 为串口下载速率。输入命令后，在超级终端的菜单中选择[传送/发送文件]，在弹出的窗口中，协议必须选择“Kermit”。 例子：`loadb 0x82000000 115200` **注意：使用 loadb，只能下载到内存中，不能直接下载到 Flash。**

tftp	从 tftp 服务器中下载文件至内存中。
	用法：tftp addr file
	说明：将 file 文件下载到地址为 addr 的内存中。
	注意：使用 tftp 时，必须先设置好网络配置，使用 setenv 配置 ipaddr、netmask、serverip 参数。
	例：
	`hisilicon > setenv ipaddr 192.168.1.1 /*设置 IP 地址*/`
	`hisilicon > setenv netmask 255.255.254.0 /*设置子网掩码*/`
	`hisilicon > setenv serverip 192.168.1.254 /*设置服务器地址*/`
	`hisilicon > tftp 0x82000000 uImage`
	说明：把 tftp 服务器（IP 为环境变量中设置的 serverip）中 **uImage** 通过 tftp 写入到内存 0x82000000 处。

cp	拷贝内存。 用法：cp [.b,.w,.l] source target count 说明：从内存地址 source 中拷贝到 target，大小为 count。实际拷贝的大小，因命令的不同而不同。 使用 cp.b，拷贝 1%count 个字节。 使用 cp.w，拷贝 2%count 个字节。 使用 cp.l，拷贝 4%count 个字节。 简单使用 cp 时，等价于 cp.1。 说明：source 和 target 是 DDR SDRAM 的地址范围。

命令	描述
go	跳转到指定地址，执行代码。 用法：go addr [arg …] 说明：执行地址 addr 处的二进制代码，可传递 arg 参数。
bootm	设置运行环境，并开始执行二进制代码。 用法：bootm [addr [arg …]] 说明：执行 addr 地址处的代码，要求二进制代码为 mkimage 处理过的二进制文件。
md	显示内存区的内容。 用法：md [.b, .w, .l] address 说明：显示地址 address 内存区内容。 使用 md.b，显示单位为 1 字节。 使用 md.w，显示单位为 2 字节。 使用 md.l，显示单位为 4 字节。 简单使用 md 时，等价于 md.l。
mm	修改内存区的内容。地址自动增加。 用法：mm [.b, .w, .l] address 说明：修改地址 address 内存区内容。 使用 mm.b，每次修改 1 字节。 使用 mm.w，每次修改 2 字节。 使用 mm.l，每次修改 4 字节。 简单使用 mm 时，等价于 mm.l。
nm	修改内存区的内容，地址不自动增加。 用法：nm [.b, .w, .l] address 说明：修改地址 address 内存区内容。 使用 nm.b，每次修改 1 字节。 使用 nm.w，每次修改 2 字节。 使用 nm.l，每次修改 4 字节。 简单使用 nm 时，等价于 nm.l。

命令	描述
mw	填充内存。 用法：mw [.b, .w, .l] address value [count] 说明：设置从地址 address 开始的 count 大小的内存为 value。 使用 mw.b，填充内存大小为 1%count 字节。 使用 mw.w，填充内存大小为 2%count 字节。 使用 mw.l，填充内存大小为 4%count 字节。 简单使用 mw 时，等价于 mw.l。 例子：**mw.b 0x82000000 FF 10000** 说明：把内存 0x82000000 开始的 0x10000 字节设为 0xFF。
cmp	比较两块内存区。 用法：cmp [.b, .w, .l] addr1 addr2 count 说明：比较地址 addr1 和地址 addr2，大小 count 的内存内容进行比较。 使用 cmp.b，比较大小为 1%count 字节。 使用 cmp.w，比较大小为 2%count 字节。 使用 cmp.l，比较的大小为 4%count 字节。 简单使用 cmp 时，等价于 cmp.l。

注：以上命令必须在同一行内输入。

在 NAND Flash 上的操作，详细信息请参见表 2-2。

表2-2 NAND Flash 命令说明

命令	描述
? nand	得到所有 nand 命令的帮助。 用法：？ nand 说明：列出 nand 命令的帮助信息。
nand info	显示所有 nand 设备的信息。 用法：nand info 举例：**nand info** 说明：本机输下出如下信息，检测到一个 nand 设备，大小为 128M，块大小为 128K。 输出：Device 0: NAND 128MiB 3,3V 8-bit, sector size 128 KiB

命令	描述
Nand device	查看具体某个 nand 设备的信息。 用法：nand device [device number] 举例：**nand device 0** 说明：显示设备 0 的说明。 输出：Device 0: NAND 128MiB 3,3V 8-bit... is now current
Nand write	用于向 NAND Flash 中写入数据。 用法：nand write mem_addr start_offset count 举例： **tftp 0x82000000 u-boot.bin**〈将 u-boot.bin 下载到内存中〉 **nand write 0x82000000 0x0 0x100000**〈将内存中起始地址为 0x82000000 中的内容写入 nand flash 偏移地址为 0 的地方，大小为 0x100000。 **说明：** 该操作是通过内存中转实现，使用该命令之前需要先使用表 2-1 中介绍的 tftp 命令,把内容下载到内存中，然后再从内存中写入 nand flash 中。
Nand write.yaffs	专用于向 NAND Flash 中下载 YAFFS2 文件系统。 用法：nand write.yaffs mem_addr start_offset fs_size 举例： **tftp 0x82000000 rootfs-FULL_REL-Flash.yaffs2** **nand write.yaffs 0x82000000 0x300000 0x702600** <这个参数是 YAFFS2 文件系统镜像的实际文件长度，注意这里是 16 进制的数> **说明：** tftp 82000000 rootfs-FULL_REL-Flash.yaffs2 扫行完之后，会打印出 Bytes transferred = 7809760 (702600 hex)。参数 fs_size 需要是 YAFFS2 文件系统的实际大小，否则会出错，其它参数含义同 Nand write。
Nand read	从 NAND Flash 中读取数据到内存中。 用法：nand read mem_addr start_offset count 举例：**nand read 0x82000000 0x100000 0x130000** 说明：将偏移地址为 0x100000 处开始，0x130000 大小的内容读取到以 0x82000000 为起始的内存中。

命令	描述
Nand erase	用来擦除 NAND Flash。 用法一：nand erase start_offset count 举例：**nand erase 0 100000** 说明：用来擦除从偏移地址 0x0 开始的 0x100000 大小空间。 用法二：**nand erase start_offset** 举例：nand erase 0x0 说明：用来擦除 0x0 开始的所有空间，即全部擦除。
Nand bad	用来探测 NAND Flash 的坏块。 用法：nand bad 举例：**nand bad** 说明：显示 NAND Flash 中的坏块信息
Nand dump	显示 NAND Flash 的数据信息。 用法：nand dump start_offset 举例：**nand dump 0x0** 说明：显示偏移地址为 0 开始的 2048Bytes 的数据信息和 64Bytes 的 OOB 信息。

说明

内存操作命令如 md，mw，mm 等对于 NAND Flash 是不适用的，命令 go 也是不适用于 NAND Flash，因为 NAND Flash 不支持 XIP（Execute In Place）。

在 SPI Flash 上的操作，详细信息请参见表 2-2。

表2-3 SPI Flash 命令说明

命令	描述
? sf	得到所有 sf 命令的帮助。 用法：？ sf 说明：列出 sf 命令的帮助信息。

命令	描述
sf probe	初始化 SPI Flash 设备。进入 uboot 后， **注意**：必须先执行该命令，然后才能对 SPI Flash 进行其他操作。 用法：sf probe 0 举例 sf probe 0。 说明：本机输下出如下信息，检测到一个 spi flash 设备，大小为 16M，块大小为 256K。 输出： Spi(cs1) ID: 0x20 0x20 0x18 0x00 0x00 0x00 Spi(cs1): Block:256KB Chip:16MB (Name:M25P128) 16384 KiB hi_sfc at 0:0 is now current device
sf write	用于向 SPI Flash 中写入数据。 用法：`sf write mem_addr start_offset count` 举例： **`tftp 0x82000000 u-boot.bin`**〈将 u-boot.bin 下载到内存中〉 **`sf write 0x82000000 0x0 0x100000`**〈将内存中起始地址为 0x82000000 中的内容写入 sf flash 偏移地址为 0 的地方，大小为 0x100000。 **说明：** 该操作是通过内存中转实现，使用该命令之前需要先使用表 2-1 中介绍的 tftp 命令,把内容下载到内存中，然后再从内存中写入 sf flash 中。
sf read	从 SPI Flash 中读取数据到内存中。 用法：sf read mem_addr start_offset count 举例：**`sf read 0x82000000 0x100000 0x130000`** 说明：将偏移地址为 0x100000 处开始，0x130000 大小的内容读取到以 0x82000000 为起始的内存中。
sf erase	用来擦除 SPI Flash。 用法：sf erase start_offset count 举例：**`sf erase 0 100000`** 说明：用来擦除从偏移地址 0x0 开始的 0x100000 大小空间。

说明

内存操作命令如 md，mw，mm 等对于 SPI Flash 是不适用的，命令 go 也是不适用于 SPI Flash，因为 SPI Flash 不支持 XIP（Execute In Place）。

2.5 U-boot 环境变量

使用 U-boot 常用命令“setenv”可以设置 U-boot 环境变量，表 2-4 列出常用环境变量及其设置格式等信息。

表2-4 U-boot 常用环境变量说明

环境变量	描述
ipaddr	设置单板的 IP 地址。 格式：**setenv ipaddr** xxx.xxx.xxx.xxx 例子：**setenv ipaddr 192.168.0.100** 说明：设置 IP 地址为 192.168.0.100。
serverip	设置服务端 IP 地址，在 tftp 中被使用。 格式：**setenv serverip** xx.xxx.xxx.xxx 例子：**setenv serverip 192.168.0.10** 说明：设置 tftp 服务器 IP 地址为 192.168.0.10。
netmask	设置子网掩码。 格式：**setenv netmask** xx.xxx.xxx.xxx 例子：**setenv netmask 255.255.255.0** 说明：设置子网掩码为 255.255.255.0。
gatewayip	设置网关。 格式：**setenv gatewayip** xxx.xxx.xxx.xxx 例子：**setenv gatewayip 192.168.0.1** 说明：设置网关 IP 地址为 192.168.0.1。
bootargs	启动 OS 时的启动参数。 格式：arg1=value1 arg2=value2 … argn=valuen 例子 1：**setenv bootargs 'mem=64M console=ttyAMA0,115200 root=/dev/mtdblock1 rootfstype=yaffs2 mtdparts=hinand:16M(boot),32M(rootfs),32M(test)'** 例子 2：setenv bootargs 'mem=64M console=ttyAMA0,115200 root=/dev/mtdblock1 rootfstype=jffs2 mtdparts=hi_sfc:5M(boot),9M(rootfs),2M(test)' 说明：传递参数，包括内存大小，根文件系统设备等。

环境变量	描述
bootcmd	设置 U-boot 自动启动及执行命令。启动延时依据 bootdelay 变量值（详见 bootdelay 参数描述），若 bootdelay 未被设置，则默认延时时间为 2 秒。 格式：cmd1; cmd2; …; cmdn 例子 1：**`setenv bootcmd 'nand read 0x82000000 100000 400000;bootm 0x82000000'`** 说明：设置启动后自动从 NAND Flash 读取偏移地址为 0x100000，长度为 0x400000 的数据到内存 0x82000000，然后执行 0x82000000 处的代码。 例子 2：**`setenv bootcmd 'sf probe 0;sf read 0x82000000 0x100000 0x400000;bootm 0x82000000'`** 说明：设置启动后自动从 SPI Flash 读取偏移地址为 0x100000，长度为 0x400000 的数据到内存 0x82000000，然后执行 0x82000000 处的代码。。 **注意：多个参数时，参数之间使用分号相隔。将整个参数字串用单引号包含起来。**
bootdelay	设置自启动延时时间。单位为秒。只有当 bootcmd 变量被设置后，该变量才有效。该变量值范围为大于等于－1 的整数。当设置为－1 时，关闭自启动的功能。 格式：value 例子 1：**`setenv bootdelay 4`** 说明：设置自启动延时 4 秒。 例子 2：**`setenv bootdelay -1`** 说明：关闭自启动功能。 提示：在延迟时间内可按任意键切换到命令行模式。 **注意：在产品开发调试阶段请勿设置延迟时间为 0。若设置，可以在启动瞬间使用 CTRL+C 中断程序而进入命令行模式。**
mdio_intf	设置 ETH 的 mdio 接口模式。Hi3518A 支持 mii 和 rgmii 两种接口模式。 格式：mode 例子 1：setenv mdio_intf rgmii 说明：设置 ETH 的 mdio 为 rgmii 模式。 例子 2：setenv mdio_intf mii 说明：设置 ETH 的 mdio 为 mii 模式。 **注意：mdio 接口模式取决于单板硬件配置。**

环境变量	描述
phyaddr	设置 ETH 的 PHY 地址。 格式：value 例子：setenv phyaddr 1 说明：设置 ETH 的 PHY 地址为 1 **注意：PHY 地址由单板硬件配置决定。**

注：以上命令必须在同一行内输入。

3 Linux 内核

3.1 内核源代码

成功安装 Hi3518A SDK 后，内核源代码已存放于 SDK 目录下的 osdrv/ kernel 目录中，用户可直接进入目录进行相关操作。另有一份内核源代码压缩包存放于 SDK 目录下的 package/目录中，用户也可以进入该目录解压缩该内核源代码压缩包，然后进行相关操作。

3.2 配置内核

注意

如果对内核和 Hi3518A 平台没有足够了解，请勿修改默认配置。但可增加需要的模块。

配置内核的操作步骤如下：

1. 手动拷贝.config 文件：

```
hisilicon$cd osdrv/kernel/linux-3.0.y
hisilicon$cp arch/arm/configs/hi3518a_full_defconfig .config
```

2. 通过“make menuconfig”进行内核配置：

```
hisilicon$make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig
```

3. 选择需要的模块。
4. 选择完毕后，保存并退出。

----结束

说明

- 编译 Hi3518C 的内核，请使用配置文件 hi3518c_full_defconfig；编译 Hi3518E 的内核，请使用配置文件 hi3518e_full_defconfig；编译 Hi3516C 的内核，请使用配置文件 hi3516c_full_defconfig。
- 文档中出现编译参数 CROSS_COMPILE 均以使用 arm-hisiv100nptl-linux-（uclibc 库）工具链为例，如果需要使用 glibc 工具链请更改编译参数 CROSS_COMPILE，具体设置如下：

  arm-hisiv100nptl-linux-工具链（uclibc 库）：CROSS_COMPILE=arm-hisiv100nptl-linux-

  arm-hisiv200-linux-工具链（glibc 库）：CROSS_COMPILE=arm-hisiv200-linux-
- 在编译 Hi3518A 的两种工具链中，uclibc 工具链支持全规格版本和小型化版本，glibc 工具链只支持全规格版本

  全规格版本：hi3518a_full_defconfig

  小型化版本：hi3518a_mini_defconfig

  文档以使用全规格版本 hi3518a_full_defconfig 为例，如果需要使用小型化版本请做相应更改。

3.3 编译内核并生成内核镜像 uImage

配置保存后，可直接输入“make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- uImage”命令编译内核生成镜像，此时需要等待几分钟。

说明

如果编译过程中出现错误，按顺序执行以下命令：

make ARCH=arm CROSS_COMPILE=arm- hisiv100nptl -linux- clean

make ARCH=arm CROSS_COMPILE=arm- hisiv100nptl -linux- menuconfig

make ARCH=arm CROSS_COMPILE=arm- hisiv100nptl -linux- uImage

4 根文件系统

4.1 根文件系统简介

Linux 的目录结构的最顶层是一个被称为“/”的根目录。系统加载 Linux 内核之后，就会挂载一个设备到根目录上。存在于这个设备中的文件系统被称为根文件系统。所有的系统命令、系统配置以及其他文件系统的挂载点都位于这个根文件系统中。

根文件系统通常存放于内存和 Flash 中，或是基于网络的文件系统。根文件系统中存放了嵌入式系统使用的所有应用程序、库以及其他需要用到的服务。图 4-1 列出了根文件系统的顶层目录。

图4-1 根文件系统顶层目录结构图

通用的 Linux 系统的根文件系统中会包括根文件系统顶层目录结构图中所有的目录，不过在嵌入式系统中，需要精简根文件系统。部分可以被忽略的目录如表 4-1 所示。

表4-1 嵌入式系统中可忽略的目录说明

目录名称	描述
/home、/mnt、/opt 和/root	所有适合提供给多用户扩展的目录，都可以被忽略。
/var 和/tmp	/var 是存放系统日志或一些服务程序的临时文件。 /tmp 是存放用户的一些临时文件，可以被忽略。
/boot	/boot 目录一般用于存放内核映像，PC 机启动时一般会从该位置加载内核，但在嵌入式系统中，为了节省空间，内核映像存在于 Flash 或网络服务器中，而不是在根文件系统中。因此也可以忽略这个目录。

注：空目录并不会增大文件系统的体积，如果没有特殊原因，建议保留这些目录。

4.2 利用 busybox 制作根文件系统

利用 busybox 制作根文件系统需要先获取 busybox 源代码，然后配置、编译和安装 busybox，操作成功后开始制作根文件系统。

4.2.1 获取 busybox 源代码

成功安装 SDK 后，busybox 完整源代码就存放在 package/目录中。要获取 busybox 源代码也可以从网站 http://www.busybox.net 下载。

4.2.2 配置 busybox

进入 busybox 所在目录，进行配置操作需要输入如下命令：

```
hisilicon$ make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig
```

busybox 的配置界面和内核配置相似，其功能选项容易理解，可以根据自己的需求选择配置。在 Busybox Settings ---> Build Options 中注意下面两个选项：

```
[*]Build BusyBox as a static binary (no shared libs)
(arm-hisiv100nptl-linux-) Cross Compiler prefix
```

其中：

- 第一个选项选择是否把 busybox 编译成静态链接的可执行文件。如果选择该选项，编译出来的 busybox 就是静态链接的，运行时不依赖于动态库，但体积较大；清除该选项将得到动态链接的 busybox，体积较小，但需要动态库的支持。
- 第二个选项是用于选择 SDK 推荐的交叉编译器，配置好后保存并退出。

欲了解 busybox 各选项含义请参考 busybox 配置帮助。

4.2.3 编译和安装 busybox

编译和安装 busybox 的具体操作如下：

```
hisilicon$ make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux-
hisilicon$ make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- install
```

编译并安装成功后，在 busybox 目录下的_install 目录下生成以下目录及文件：

```
drwxr-xr-x  2 linux  linux  4096 2005-04-22 11:01 bin
lrwxrwxrwx  1 linux  linux  11 2005-04-22 11:01 linuxrc->bin/busybox
drwxr-xr-x  2 linux  linux  4096 2005-04-22 11:01 sbin
drwxr-xr-x  4 linux  linux  4096 2005-04-22 11:01 usr
```

4.2.4 制作根文件系统

成功安装 SDK 后，在 osdrv/rootfs_scripts/目录中存放已制作好的根文件系统压缩包 rootfs.tgz。

用户如有需要可在 busybox 的基础上制作根文件系统。

制作根文件系统的具体操作步骤如下：

1. hisilicon$mkdir rootfs

   hisilicon$cd rootfs

   hisilicon$cp –R package/osdrv/ busybox/busybox-1.16.1/_intsall/* .

   hisilicon$**mkdir etc dev lib tmp var mnt home proc**

2. 配置 etc、lib、dev 目录的必需文件。

   a. etc 目录可参考系统/etc 下的文件。其中最主要的文件包括 inittab、fstab、init.d/rcS 文件等，这些文件最好从 busybox 的 examples 目录下拷贝过来，根据需要自行修改。

   b. dev 目录下的设备文件，可以直接从系统中拷贝过来或者使用 mknod 命令生成需要的设备文件。拷贝文件时请使用 cp –R file。

   c. lib 目录是存放应用程序所需要的库文件，请根据应用程序需要拷贝相应的库文件。

   ----结束

完成以上两个步骤，一个完整的根文件系统就生成了。

说明

SDK 软件包中已经包括配置好的完整的根文件系统，如果无特别需求，可直接使用。要添加自己开发的应用程序，只需将应用程序和相应的库文件拷贝到根文件系统的对应目录即可。

4.3 文件系统简介

嵌入式系统中常用文件系统包括有 SquashFS、JFFS2、NFS 以及 YAFFS2。它们的特点如下：

- SquashFS 和 JFFS2 具有好的空间特性，很适合嵌入式产品应用。
- SquashFS 为只读文件系统。
- JFFS2 为可读写文件系统。
- NFS 文件系统适用于开发初期的调试阶段。
- YAFFS2 文件系统只用于 NAND Flash。

4.3.1 JFFS2

JFFS2 是 RedHat 的 David Woodhouse 在 JFFS 基础上改进的文件系统，是用于微型嵌入式设备的原始闪存芯片的实际文件系统。JFFS2 文件系统是日志结构化的可读写的文件系统。

JFFS2 的优缺点如下：

- 优点

  使用了压缩的文件格式。最重要的特性是可读写操作。

- 缺点

  JFFS2 文件系统挂载时需要扫描整个 JFFS2 文件系统，因此当 JFFS2 文件系统分区增大时，挂载时间也会相应的变长。使用 JFFS2 格式可能带来少量的 Flash 空间的浪费。这主要是由于日志文件的过度开销和用于回收系统的无用存储单元，浪费的空间大小大致是若干个数据段。JFFS2 的另一缺点是当文件系统已满或接近满时，JFFS2 运行速度会迅速降低。这是因为垃圾收集的问题。

加载 JFFS2 文件系统时的步骤如下：

1. 扫描整个芯片，对日志节点进行校验，并且将日志节点全部装入内存缓存。
2. 对所有日志节点进行整理，抽取有效的节点并整理出文件目录信息。
3. 找出文件系统中无效节点并且将它们删除。
4. 最后整理内存中的信息，将加载到缓存中的无效节点释放。

----结束

由此可以看出虽然这样能有效地提高系统的可靠性，但是在一定程度上降低了系统的速度。尤其对于较大的闪存芯片，加载过程会更慢。

为了使内核支持 JFFS2 文件系统，必须在编译内核时把 JFFS2 的选项加入（我们发布的内核默认已经加入了支持）。在 make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig 后，进入“File systems”，选择“miscellaneous filesystems”，最后选中其中的“Journalling Flash File System v2 (JFFS2) support”选项（SDK 里面提供的内核默认已经选择了该文件系统的支持）。

JFFS2 的制作方法为：

```
hisilicon$ mkfs.jffs2 -d ./rootfs -l -e 0x20000 -o  jffs2-root.img
```

其中，mkfs.jffs2 工具可以从互联网中下载，也可以在 SDK 包中找到。rootfs 为之前已经制作好的根文件系统。参数说明如表 4-2 所示。

表4-2 JFFS2 参数表

参数	说明
d	指定根文件系统
l	little-endian 小端模式
e	Flash 的块大小
o	输出映像文件

4.3.2 NFS

使用 SquashFS 和 JFFS2 时，需要先将根文件系统映像烧入 Flash，系统启动时会从 Flash 中加载。但是在系统开发或移植的初期，需要经常修改或者添加应用程序。每修改一次就需要重新烧入一次，这样做不仅耗费时间，而且对 Flash 的寿命会有影响。

NFS 是一种分布式的文件系统，用于共享文件和打印机。它允许用户调用挂载远端的文件系统或设备来实现共享，使用方式与挂载本机的文件系统一样。NFS 使用“客户－服务器”模型。在这种模型中，服务器输出需要共享的目录，客户可通过网络挂载这些目录并访问其中的文件。

使用 NFS 作为挂载的文件系统，内核会根据预先设置好的内核命令参数挂载一个 NFS sever 中输出的目录作为其根目录。这个过程不需要任何对 Flash 的操作，修改应用程序完全在 Linux 服务器中进行，非常适于开发初期的调试阶段。

在 Linux 服务器配置 NFS 文件系统的方法为：编辑/etc/exports 配置文件，添加路径及参数，然后执行/etc/init.d/ nfs start 启动 NFS 服务。

以上操作必须超级用户完成，且导出的目录必须是绝对路径。如果 NFS 服务已经开启，在配置文件后只需重新启动 NFS 服务，即/etc/init.d/ nfs restart。

在 Linux 服务器上配置好 NFS 根文件系统后，在单板侧挂载 NFS 文件系统，具体操作如下：

```
ifconfig eth0 hw ether 00:10:85:18:01:84            /*配置MAC地址*/
ifconfig eth0 10.85.180.184 netmask 255.255.254.0   /*配置IP地址和子网掩码*/
route add default gw 10.85.180.1                    /*配置默认网关*/
mount -t nfs -o nolock 10.85.180.133:/home/c54122/glibc-nfs /mnt     /*挂载NFS目录至JFFS2文件系统的mnt目录下*/
```

4.3.3 YAFFS2

YAFFS2 是专门为 NAND Flash 设计的嵌入式文件系统。它是日志结构的文件系统，提供了损耗平衡和掉电保护，可以有效地避免意外掉电对文件系统一致性和完整性的影响。

YAFFS2 的优缺点如下：

- 优点
  - 专门针对 NAND Flash，软件结构得到优化，速度快。
  - 使用硬件的 spare area 区域存储文件组织信息，启动时只需扫描组织信息，启动比较快。
  - 采用多策略垃圾回收算法，能够提高垃圾回收的效率和公平性，达到损耗平衡的目的。
- 缺点

  没有采用压缩的文件格式。当包含的内容相同时，YAFFS2 镜像文件要比 jffs2 镜像文件大。

YAFFS2 文件系统在 SDK 中作为一个模块提供。只需在 YAFFS2 代码中的 Makefile 中加入所依赖的内核代码路径，进行编译，即可生成 YAFFS2 文件系统模块。

YAFFS2 镜像文件的制作和 SquashFS 相同，即通过工具制作，只需简单的几个参数，具体如下：

```
hisilicon$ mkyaffs2image  ./rootfs ./yaffs2-root.img pagesize ecctype
```

其中，rootfs 是之前已经制作好的根文件系统，yaffs2-root.img 是生成的 YAFFS2 文件系统镜像文件，pagesize 是单板上焊接 NAND Flash 器件的页大小，ecctype 是单板上焊接 NAND Flash 器件的 ecc 类型。

4.3.4 SquashFS

SquashFS 是另一种可用于 flash 设备的 Linux 只读文件系统。SquashFS 具有极高的压缩率，数据(data),节点(inode)和目录(directories)都被压缩。常用于存储介质很有限的场景。

SquashFS 保存了全部的 32 位 UID/GIDS 和文件的创建时间，支持多达 4G 的文件系统，SquashFS 使用简单，响应速度快。

SquashFS 是一种新的文件系统，它对 CramFS 的特点作了进一步的改进，并突破了 CramFS 的一些限制。其优点如下：

- SquashFS 具有更高的压缩率
- SquashFS 的响应速度更快
- CramFS 文件系统支持的最大文件为 16M，而 SquashFS 多达 4G
- CramFS 文件系统大小限制略大于 256M，而 SquashFS 多达 4G

如果要使用 SquashFS，需要在内核中增加相应的配置选项：make ARCH=arm CROSS_COMPILE=arm-hisiv100nptl-linux- menuconfig；然后选择下面选项

File systems ---->

[*] Miscellaneous filesystems ---->

<*> SquashFS 4.0 - Squashed file system support

mksquashfs 是用来制作 SquashFS 文件系统映象的工具。通过这个工具处理已经制作好的根文件系统，就可以生成 SquashFS 文件系统的映象，制作 SquashFS 的方法：./mksquashfs ./rootfs ./rootfs.squashfs.img -b 256K，其中：

- rootfs 是已经制作好的根文件系统。
- rootfs.squashfs.img 是生成的 SquashFS 文件系统映像文件。
- -b 256K 指定 SquashFS 文件系统的块大小为 256K。

5 烧写内核和根文件系统

5.1 存储器地址空间

Hi3518A DMEB 包含 DDR 存储器、SPI Flash 存储器和 NAND Flash 存储器。DDR 的地址空间从 0x80000000 开始；SPI Flash 的地址空间从 0x58000000 开始，NAND Flash 的地址空间从 0x50000000 开始。各种 Flash 分配方式相同，下面以 SPI Flash 为例进行说明。具体大小随单板不同，可从单板硬件手册中获取。

Flash 的使用有特殊要求，以 16MB 的 SPI Flash 为例：

- 0x0～0x100000 保留空间，存放 U-boot。
- 0x100000～0x500000 保留空间，供存放内核。
- 0x500000～0xE00000 保留空间，供存放文件系统。
- 其余则保留或有其他用途。

5.2 通过网口烧写

通过网口烧写内核和根文件系统，首先需要进行参数设置和建立 tftp 服务，然后才能下载内核和根文件系统。

5.2.1 参数设置和建立 tftp 服务

用普通网线连接 Hi3518A DMEB 的 ETH 网口，然后在 U-boot 中设置相关参数。U-boot 只支持 tftp 协议。设置参数的具体操作如下所示：

```
hisilicon#setenv serverip 10.85.180.211     /*设置服务器端的IP地址，可根据需要具体设定*/
hisilicon#setenv ipaddr 10.85.180.130       /*设置Hi3518A DMEB板的IP地址*/
hisilicon#setenv netmask 255.255.254.0      /*设置netmask*/
hisilicon#setenv gatewayip 10.85.180.1      /*设置网关*/
hisilicon#saveenv
hisilicon# ping 10.85.77.69                 /*用来判断网络是否正常*/
```

U-boot 不支持广播包的接收。U-boot 支持向外发 ping 包，并能接收 ping 包的响应包。可通过单板 ping 主机来验证网络是否连接正常。上述最后一个操作中，返回 `host 10.85.77.69 is alive` 表示网络工作正常；显示 `ping failed; host 10.85.77.69 is not alive`，说明网络不正常，需要重新检查网络设置。

另外还需要在 Windows 工作台或者 Linux 服务器中建立 tftp 服务，建议在 Windows 工作台上建立 tftp 服务器，简单方便。

5.2.2 下载内核

在 Hi3518A DMEB 上下载内核的操作如下所示。

将内核写入 SPI Flash：

hisilicon#sf probe 0　/*对 SPI Flash 进行初始化设置*/

```
hisilicon#sf erase 0x100000 0x300000 /*在进行FLASH写入之前必须先手动擦除FLASH，否则写入FLASH的数据错误*/
hisilicon#tftp 0x82000000 uImage /*将tftp服务器上的uImage文件下载到0x82000000的位置*/
```

正常的下载过程超级终端中显示的信息如下所示：

```
MAC:   00-10-85-18-01-30
TFTP from server 10.85.180.211; our IP address is 10.85.180.130
Download Filename ' uImage'.
Download to address: 0x82000000
Downloading: %T%T%#     [ Connected ]
################################
        0.988 MB download ok.
Bytes transferred = 1012660 (f73b4 hex)
hisilicon#sf write 0x82000000 0x100000 0x300000  /*将0x82000000的数据写入SPI Flash的0x100000地址处，写入长度为0x300000*/
```

将内核写入NAND Flash：

```
hisilicon#nand erase 0x100000 0xf00000 /*在进行FLASH写入之前必须先手动擦除FLASH，否则写入FLASH的数据错误*/
hisilicon#tftp 0x82000000 uImage /*将tftp服务器上的uImage文件下载到0x82000000的位置*/
hisilicon#nand write 0x82000000 0x100000 0xf00000  /*将0x82000000的数据写入NAND Flash的0x100000地址处，写入长度为0xf00000*/
```

5.2.3 下载根文件系统

文件系统有 SPI Flash 和 NAND Flash 的文件系统。

下载根文件系统的操作以 SPI Flash 为例如下所示。

```
hisilicon#sf probe 0  /*对 SPI Flash 进行初始化设置*/
```

```
hisilicon#sf erase 0x500000 0x900000 /*首先擦除 FLASH 的文件系统分区，FLASH 分区信息参见后面的启动参数设置*/
hisilicon#tftp 0x82000000 rootfs-FULL_REL.jffs2  /*将 rootfs-FULL_REL.jffs2 文件下载到 0x82000000*/
```

正常的下载过程超级终端中显示的信息如下所示：

```
MAC:   00-10-85-18-01-30
TFTP from server 10.85.180.211; our IP address is 10.85.180.130
Download Filename 'rootfs-FULL_REL.jffs2'.
Download to address: 0x80800000
Downloading: %# [ Connected ]
#################################       [ 1.000 MB]
#################################       [ 2.000 MB]
#################################       [ 3.000 MB]
#################################       [ 4.000 MB]
#################################       [ 5.000 MB]
#################################       [ 6.000 MB]
###################
        6.591 MB download ok.
Bytes transferred = 6897136 (693df0 hex)
hisilicon#sf write 0x82000000 0x500000 0x900000
```

由于 SPI Flash 的写操作速度较慢，如果下载的文件较大，则需要花费一定的时间，等到重新回到“hisilicon#”的提示符，表示下载完成。

下载根文件系统的操作以 NAND Flash 为例如下所示。

```
hisilicon#nand erase 0x1000000 0x2000000 /*首先擦除 NAND FLASH 的文件系统分区，NAND FLASH 分区信息参见后面的启动参数设置*/
hisilicon # tftp 82000000 2k1b.yaffs2  /*将 2k1b.yaffs2 文件下载到 0x82000000*/
```

正常的下载过程超级终端中显示的信息如下所示：

```
miiphy_register: non unique device name '0:01'
MAC:   00-00-23-34-45-66
TFTP from server 10.85.181.114; our IP address is 10.85.180.193
Download Filename '2k1b.yaffs2'.
Download to address: 0x82000000
Downloading: #################################################
done
Bytes transferred = 9928512 (977f40 hex)
hisilicon # nand write.yaffs 82000000 1000000 977F40

NAND write: device 0 offset 0x1000000, size 0x977f40
pure data length is 9627648, len_incl_bad is 9699328

 9928512 bytes written: OK
```

注意：上面 977f40 这个参数是 **YAFFS2** 文件系统镜像的实际文件长度，并且这里是 **16** 进制的数。当重新回到“hisilicon#”的提示符，表示下载完成。

5.3 通过串口烧写

通过串口烧写内核和根文件系统，首先需要在 Windows 工作台和 Hi3518A DMEB 之间通过串口连接，然后才能下载内核和根文件系统。

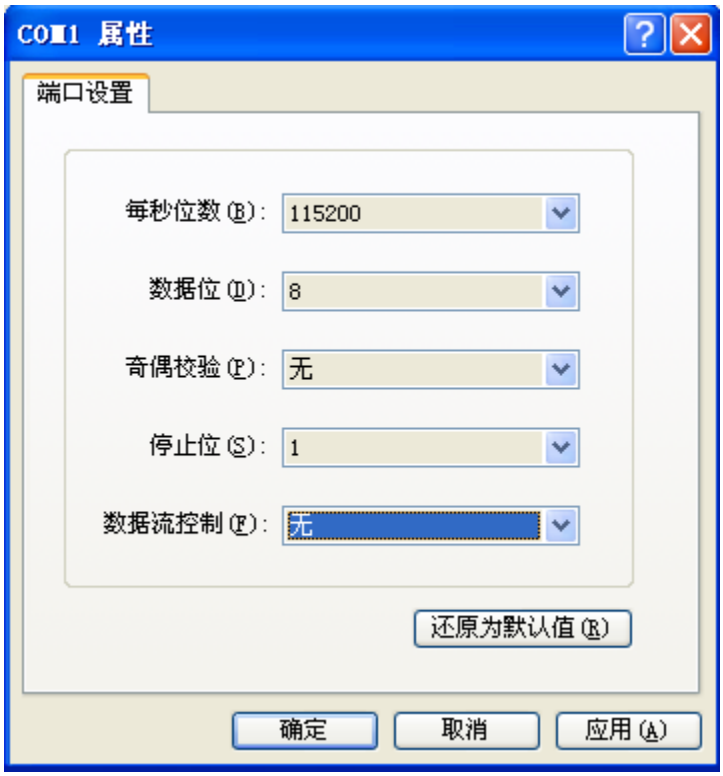
5.3.1 连接设备

连接设备的步骤如下：

1. 使用串口线（DB9 接口）连接 Windows 工作台的 COM1（也可以是其他串口，这里假设使用 COM1）和 Hi3518A DMEB 的 COM。
2. 启动 Windows 工作台的超级终端软件，设置 COM1 的参数如图 5-1 所示。
3. 启动 Hi3518A DMEB，系统进入 U-boot 命令行操作界面，表示系统工作正常，可进行下载或其他操作。

----结束

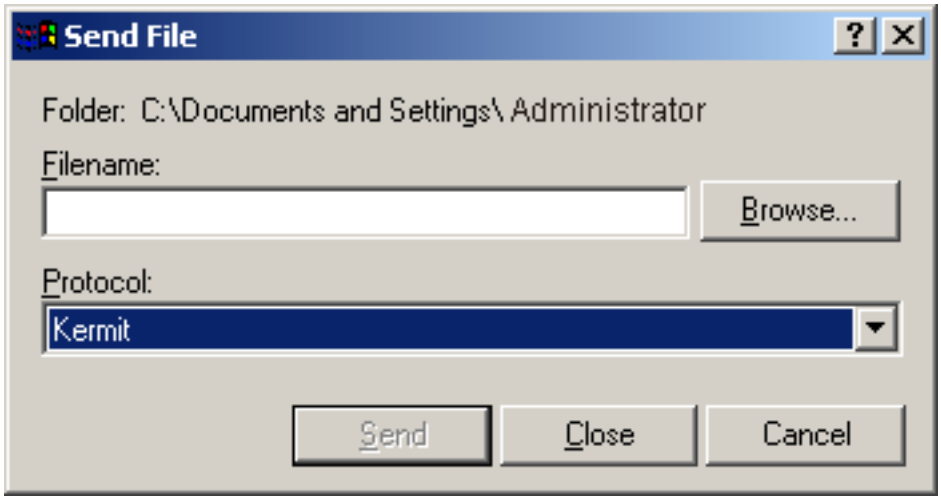
图5-1 串口设置



5.3.2 下载内核

在超级终端的 U-boot 命令行中输入 **loadb 0x82000000**（存放内核的内存地址），打开菜单“传输”下的“发送文件”，弹出对话框，如图 5-2 所示，选择“Protocol>Kermit”，在“Filename”中选择内核文件。

图5-2 发送文件窗口



等待下载完成后，使用 U-boot 的 sf 命令系列将内核从内存拷贝到 SPI Flash 中，这里以 SPI Flash 操作为例：

```
hisilicon#sf probe 0    /*对SPI Flash进行初始化设置*/
hisilicon#sf erase 0x100000 0x300000          /*擦除Flash*/
hisilicon#sf write 0x82000000 0x100000 0x300000    /*将内核写入到SPI Flash偏移地址为0x100000位置*/
```

5.3.3 下载根文件系统

下载根文件系统和下载内核的操作方法相同，具体步骤如下：

1. 在超级终端的 U-boot 命令行中输入 **loadb 0x82000000**（存放根文件系统映像文件的内存地址）。
2. 选择“传输>发送文件”，弹出对话框，在“Protocol”的下拉列表中选择“Kermit”选项，在“Filename”中选择根文件系统映像文件（在“4 根文件系统”中已经制作好的“rootfs-FULL_REL.jffs2”或者“yaffs2-root.img”）。
3. 等待下载完成后，使用 U-boot 的 sf 命令系列将文件从内存拷贝到 Flash 中。

   具体操作方法以 SPI Flash 为例如下所示：

```
hisilicon#sf probe 0    /*对SPI Flash进行初始化设置*/
hisilicon#sf erase 0x500000 0x900000          /*擦除Flash*/
hisilicon#sf write 0x82000000 0x500000 0x900000    /*将文件系统写入到SPI Flash偏移地址为0x500000位置*/
```

----结束

说明

使用串口下载速度很慢但操作简单，适合下载小文件，而通过网口下载速度很快，提高工作效率。建议使用网口下载。

6 启动 Linux

6.1 设置启动参数以及自动启动方式

从 U-boot 引导内核，需要给内核传递参数，包括内存大小、根文件系统挂载设备等。根据根文件系统类型不同，设置也相应不同，这里仅提供一种参考设置。

各参数的含义如下：

- mem：设置操作系统内存大小。以上设置 mem=64M 表示分配给操作系统内存为 64M。
- console：设置控制台设备。格式为 console=ttyAMA0,115200 表示控制台为串口 0，波特率 115200。
- root：设置根文件系统挂载设备。格式为 root= /dev/mtdblock1 表示从 Flash 第 1 个分区挂载（Flash 分区编号从 0 开始）。
- rootfstype：设置挂载文件系统类型。
- mtdparts：Flash 分区描述，格式为 mtdparts=hi_sfc:5M(boot),9M(rootfs)表示有两个分区，分区 0 大小为 5M 用于内核启动，分区 1 大小为 9M 用于文件系统。

注意

以下参数设置，必须在同一行中输入。

JFFS2

根文件系统类型为 JFFS2 时，设置如下：

```
hisilicon# setenv bootargs 'mem=64M console=ttyAMA0,115200
root=/dev/mtdblock1 rootfstype=jffs2 mtdparts=hi_sfc:5M(boot),9M(rootfs)'
setenv bootcmd 'sf probe 0;sf read 0x82000000 0x100000 0x400000;bootm
0x82000000'  /*设置自启动命令参数*/
hisilicon#setenv bootdelay 2                /*设置启动延时为2秒*/
hisilicon#saveenv
```

YAFFS2

根文件系统类型为 YAFFS2 时，设置如下：

```
hisilicon# setenv bootargs 'mem=64M console=ttyAMA0,115200
root=/dev/mtdblock1 rootfstype=yaffs2
mtdparts=hinand:16M(boot),32M(rootfs),32M(test)'
setenv bootcmd 'nand read 0x82000000 100000 400000;bootm 0x82000000'  /*设置自启动命令参数*/
hisilicon#setenv bootdelay 2                /*设置启动延时为2秒*/
hisilicon#saveenv
```

在 U-boot 命令行中输入 reset 即可。

```
hisilicon#reset /*复位单板，自动启动Linux*/
```

SquashFS

根文件系统类型为 SquashFS 时，设置如下：

```
hisilicon# setenv bootargs ‘mem=64M console=ttyAMA0,115200
root=/dev/mtdblock1 rootfstype=squashfs
mtdparts=hi_sfc:5M(boot),9M(rootfs)’
setenv bootcmd 'sf probe 0;sf read 0x82000000 0x100000 0x400000;bootm 0x82000000'  /*设置自启动命令参数*/
hisilicon#setenv bootdelay 2                /*设置启动延时为2秒*/
hisilicon#saveenv
```

7 应用程序开发简介

7.1 编写代码

用户可根据个人习惯选择代码编写工具。通常在 Windows 环境下使用 Source Insight，在 Linux 环境下使用 Vim+ctags+cscope，功能也相当强大。

7.2 运行应用程序

要运行编译好的应用程序，首先需要将其添加到目标机中，必须完成以下工作：

- 将应用程序和需要的库文件（如果有）等添加到目标机的根文件系统相应的目录中。通常将应用程序放到/bin 目录里，库文件放到/lib 目录里，配置文件则放到/etc 目录里。
- 制作包含新应用程序的根文件系统。

说明

如果执行应用程序，需要读写文件系统操作。请选择 YAFFS2、JFFS2 文件系统。

在调试阶段推荐使用 NFS 文件系统，可以省去重新制作根文件系统和烧写工作。设置和启动 NFS 服务（请参见“4.3.2 NFS”），然后将 NFS 目录挂载到 YAFFS2、JFFS2 文件系统目录中，操作方法如下：

```
mount -t nfs -o nolock serverip:path /mnt
```

其中 serverip 表示 NFS 目录所在服务器的 ip，path 表示 NFS 目录在服务器上的路径，以后只需要简单的拷贝应用程序到 NFS 系统目录中，就可以在目标机里运行。

如果需要制作 SquashFS、YAFFS2 或 JFFS2 文件系统，制作相应的文件系统（请参见“4.3 文件系统简介”），然后烧写根文件系统到 Flash 指定位置（请参见“5 烧写内核和根文件系统”），并设置相应的启动参数。同样，启动 Linux 后便可运行新的应用程序。

说明

如果新添加的应用程序需要系统启动后自动运行，请编辑/etc/init.d/rcS 文件，添加需要启动的用程序路径。

A 缩略语

A

ADS	ARM Development Suite	ARM 开发工具套件
ARM	Advanced RISC Machine	ARM 公司指令集

C

SQUASHFS	Squash Filesystem	Squash 文件系统

D

DMS	Digital Media Solution	媒体解决方案平台

E

ELF	Executable and Linkable Format	可执行连接格式文件

G

GCC	GNU Complier Collection	GNU 编译器集合
GDB	GNU Debugger	GNU 调试器
GNU	GNU’s Not UNIX	GNU

I

IP	Internet Porotocol	Internet 协议

J

JFFS2	Journalling FLASH File System v2	一种 Flash 文件系统

JTAG	Joint Test Action Group	联合测试行动组
N		
NFS	Network File System	网络文件系统
P		
PC	Personal Computer	个人计算机
S		
SquashFS	Squash Files system	Squash 文件系统
Y		
YAFFS2	Yet Another Flash File System	一直专门针对 NAND 闪存设计的文件系统