

电 子 科 技 大 学

嵌入式智能计算研究团队

# 珊瑚-I 入门引导

ACORAL-I GUIDE MANUAL



版本号      1.2

---

## Revision History

版本号	内容	日期	负责人
0.1	开始编写, 修改 latex 模板, 确定大纲	2022.05.15	王彬浩
1.0	第一版 aCoral 入门引导编写完成	2022.06.12	王彬浩
1.1	增加 Ubuntu 和编译器、dnw 工具的 下载链接	2022.06.14	王彬浩
1.2	修正配置环境时的操作	2022.06.14	王彬浩

## 目 录

第一章 概述.....	1
1.1 珊瑚（aCoral）简介 .....	1
1.2 aCoral 项目成员 .....	1
1.2.1 aCoral 早期版本项目成员 .....	1
1.2.2 aCoral-I 项目成员.....	3
第二章 aCoral 结构.....	4
2.1 aCoral 系统结构 .....	4
2.2 aCoral 文件结构 .....	5
第三章 使用介绍.....	7
3.1 配置编译环境 .....	7
3.2 aCoral 内核下载.....	8
第四章 相关资料.....	14

## 第一章 概述

### 1.1 珊瑚（aCoral）简介

珊瑚（aCoral）是电子科技大学信息与软件工程学院嵌入式智能计算研究团队开发的一款嵌入式实时操作系统,具有开源、高可配、高扩展性的特点。

珊瑚（aCoral）目前拥有单核（aCoral-I）和多核（aCoral-II）两个版本。本仓库中的文档将介绍珊瑚操作系统的单核版本 aCoral-I，使用的硬件平台为 mini2440。出于方便的目的，后续将单核版本的珊瑚简称为 aCoral。单核版本的珊瑚（aCoral-I）对于主流的开发平台都有支持，像 s3c2440,s3c2410,s3c44b0,lpc2313,lpc2200,stm3210。

aCoral 支持多线程模式，其最小配置生成的代码为 7K 左右，而配置文件系统、轻型 TCP/IP、GUI 后生成的代码仅有 300K 左右。

嵌入式操作系统一般都是实时的，但是如何做到强实时是一个很棘手的问题，为强实时计算密集型应用（航空电子、舰载电子 ,,,）提供可靠运行支持是 aCoral 开发的强力主线。目前 aCoral 提供了强实时内核机制（优先级位图法、优先级天花板协议、差分时间链、最多关中断时间）。与此同时，aCoral 还提供了强实时调度策略：RM 调度算法，强实时确保策略也正在研究中。

aCoral 会像珊瑚一样成长.....

### 1.2 aCoral 项目成员

#### 1.2.1 aCoral 早期版本项目成员

成员姓名	主要贡献
廖勇	项目创建人及总负责人
申建晶	内核框架设计及实现，GUI 系统
闫志强	内核线程交互开发及 TCP/IP 协议栈
孔帅帅	多核支持及中断系统
高攀	文件系统开发，H.264 在 ARM11 的多核优化
陈旭东	多核调度、实时性确保、多核实时控制
刘晓翔	开发环境及驱动模块

杨茂林	强实时调度算法研究及实现
张国梁	操作系统移植
王小溪	系统测试及性能确保、应用程序
魏守峰	系统测试及性能确保、应用程序
任艳伟	内核实时性确保
程潜	内核实时性确保
程勇明	内核实时性确保
汪琳玫	功耗管理
周强	驱动模型设计及自有图形系统开发
许斌	操作系统配置工具开发
袁霞	调度算法
江维	可信调度、功耗管理
李波	操作系统移植
张海斌	多核中断支持
钟太聪	多核调试器
彭东脉	多核调试器
Mugundhan balaji	H.264 多核支持
Subhajit Banerjee Purnapatra	多核调试器
孙康	内存管理
王云飞	内存管理
郭治姣	操作系统移植
韩炫	操作系统移植
文秀春	操作系统移植
刘坚	操作系统移植
龚俊儒	操作系统移植
李天华	操作系统移植
郭文生	多核形式化验证
刘洋	应用程序
熊光泽	技术指导

桑楠	技术指导
雷航	技术指导
罗蕾	技术指导
李允	技术指导
陈丽蓉	技术指导

### 1.2.2 aCoral-I 项目成员

成员姓名	主要贡献
杨茂林	项目负责人
王彬浩	代码重构、文档编写、应用开发

## 第二章 aCoral 结构

### 2.1 aCoral 系统结构

aCoral 由内核（kernel）和外围模块（Peripheral）两大部分组成。其中内核又包含中断管理系统、内存管理系统、线程管理系统和线程交互系统；外围模块包括驱动管理、图形用户界面（GUI）、文件系统和网络模块（Net）。如图 2-1 所示。

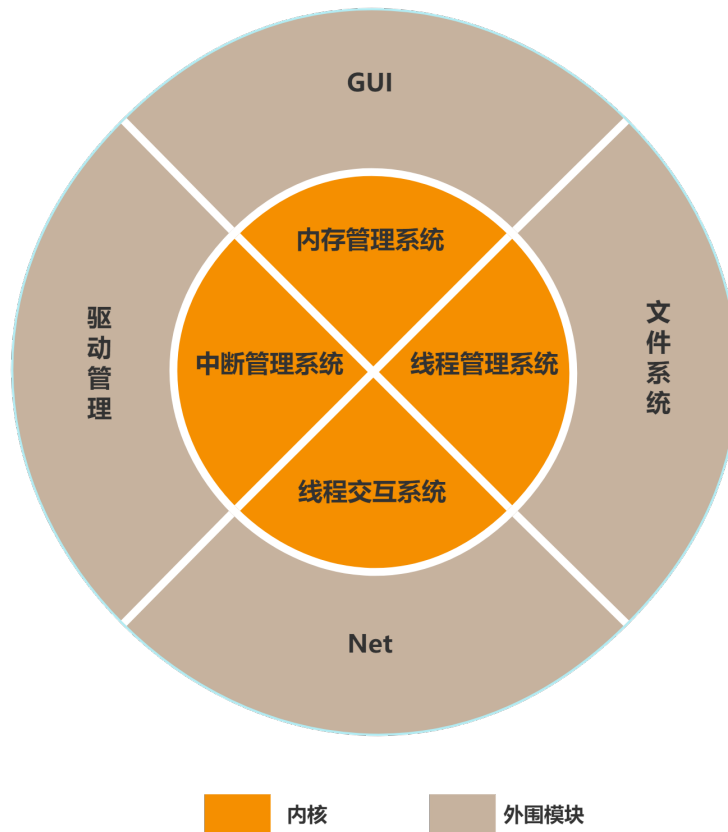


图 2-1 aCoral 系统结构

内核当中，中断管理系统负责响应并处理来自外部和内部的所有中断（异常），例如时钟中断、按键中断等；内存管理系统负责对 mini2440 上的 SDRAM 内存进行管理，包括内存的分配、回收算法的实现；线程管理系统包括线程调度机制和线程调度策略两个部分，负责创建、挂起、杀死线程等操作以及按照何种

策略来调度线程；线程交互系统包括互斥量、信号量、邮箱、消息队列等线程间交互机制。

## 2.2 aCoral 文件结构

aCoral 的一大特点是可配置性，这就要求好的系统文件结构。aCoral 内核主要由 7 个文件夹组成：

(1)kernel, 内核文件夹. 该文件夹下又有两个文件夹：

- i. include: 内核模块的头文件目录
- ii. src: 内核模块的源码目录

(2)hal (Hardware Abstract Layer), 硬件抽象层文件夹。这里存放各种开发板硬件相关的底层代码。

(3)include,aCoral 一些重要配置头文件。

(4)driver, 驱动文件夹。此文件夹存放系统的驱动程序：

- i. src, 这里存放平台无关驱动模型实现，比如驱动模型，sd 卡驱动模型。
- ii. include, 这里存放平台无关驱动模型的头文件，比如 screen 设备的信息结构，触摸屏设备的信息结构。
- iii. 开发板相关驱动文件夹，s3c2440,s3c2410, 每个文件夹下又各自包含 include,src 文件夹。

(5)plugin, 项目扩展插件目录，比如文件系统，图形系统，TCP/IP 协议栈等等。

- i. src, 扩展插件的公共源码。
- ii. include , 扩展插件的公共头文件。
- iii. 具体的扩展插件文件夹。

(6)lib, 库目录。

- i. src, 源码目录。
- ii. include, 头文件

(7)user, 用户程序目录。

i. src, 源码目录。user.c 中的 user\_main 是用户程序的入口函数，大家可以在这个函数里添加自己的应用。

- ii. include, 头文件

(8)test, 测试文件目录。主用用于内部测试。

- i. src, 源码目录。
- ii. include, 头文件

除了这些文件夹以外，aCoral 根目录下还有一些重要的文件，Makefile、



.config(menuconfig 自动生成)……这些文件都有十分重要的作用，可以有空自行学习。另外那些 acoral 打头的文件则是在编译过程中自动生成的一些辅助文件，有助于开发人员 debug 和开发应用。

## 第三章 使用介绍

在这一章中，手册使用的操作系统为 Ubuntu 18.04.5 LTS，编译交叉工具链为 arm-2010q1。其它版本的操作系统和编译器可以自行测试。点击 [工具链接](#) 下载所需工具。

### 3.1 配置编译环境

(1) 第一步，将编译器加入系统环境变量。在终端输入

```
1 gedit /etc/profile
```

在打开的文件最后另起一行输入（很重要，请务必确认内容正确）

```
1 export PATH=$PATH:xxx/arm-2010q1/bin
```

其中“xxx”为交叉编译链文件所在路径。然后保存文件，在终端输入（再次提醒，这里很重要，一旦环境变量出错影响到系统，后果很严重）

```
1 source /etc/profile
```

更新环境变量，再手动重启，最后在终端输入

```
1 arm-none-eabi-gcc -v
```

查看版本号，确定交叉编译工具是否安装成功。

如果出现“no such file or directory”的错误，则是因为 32 位编译器不能在 64 位系统中运行，需要安装 32 位库。在终端中输入

```
1 apt-get install lib32ncurses5 lib32z1
```

安装完成后，再次查看版本号即可。

(2) 第二步，修改编译所需的编译器。修改 aCoral 根目录下的 Makefile 文件中的交叉编译路径 CROSS\_COMPILE 为

```
1 xxx/arm-2010q1/bin/arm-none-eabi-
```

其中“xxx”为交叉编译链文件所在路径。

(3) 第三步，编译。进入 aCoral 根目录，在终端输入

```
1 make
```

等待编译完成之后，就会得到我们要下载的 aCoral 镜像文件 acoral.bin 以及一些辅助文件。

## 3.2 aCoral 内核下载

在得到 `acoral.bin` 镜像文件后，我们就可以将其下载到开发板上了。理论上我们可以直接把 `acoral.bin` 烧写到开发板的 `nor flash` 或者 `nand flash` 上启动，`aCoral` 可以自我引导，即将自己复制到 `sdram` 内存中执行。但是这有一个问题，烧写到 `nor flash` 或者 `nand flash` 的速度是很慢的，如果我们每次在修改 `aCoral` 的源码后，直接烧写到开发板上进行调试，就会浪费大量时间在等待烧写完成上。

对于这个问题，我们有一种解决方案，就是在 `nor flash` 或者 `nand flash` 中烧写一个 `bootloader`，每次修改源码编译得到新镜像后，使用 `bootloader`，将新编译得到的内核镜像直接复制到 `sdram` 中执行，这样就可以节省很多时间。本手册中，`bootloader` 我们使用友善之臂开发的 `supervivi`。关于 `supervivi` 如何烧写进 `flash`，请自行在网上查找资料。这里我们默认已经在 `nor flash` 中烧写好了 `supervivi`。接下来就可以正式开始下载 `aCoral`。

(1) 第一步，接线。依次从左到右为 USB 下载线、串口线、电源线，如图3-1。

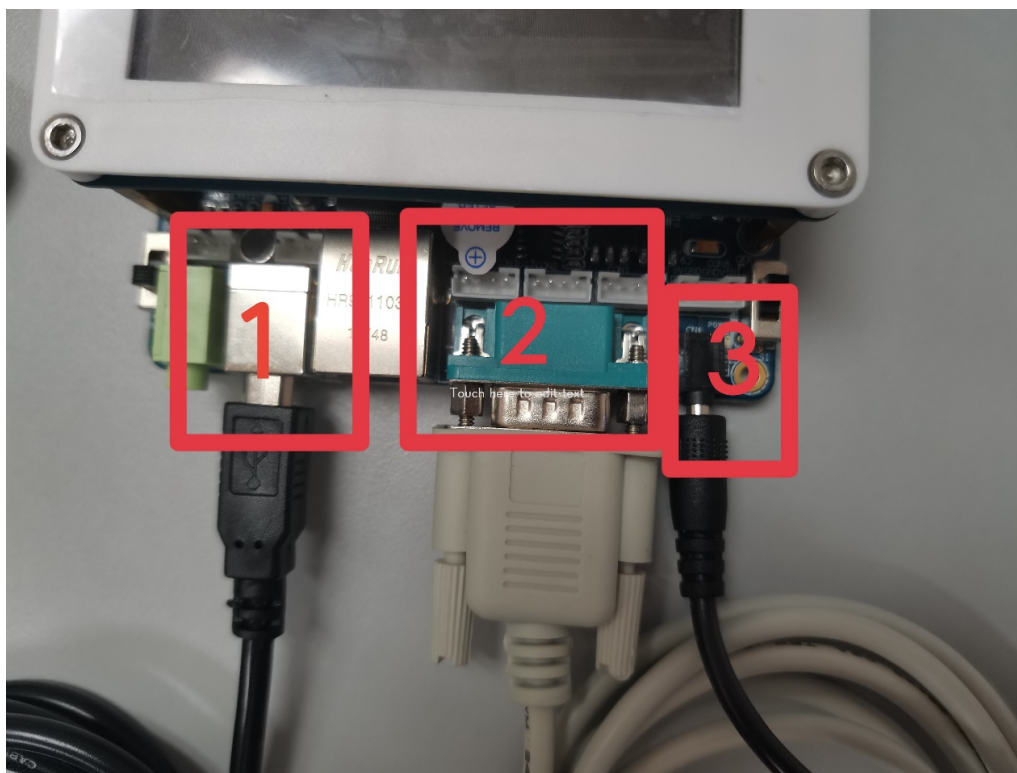


图 3-1 接线示意图

大家电脑上肯定没有串口了，所以需要接一个 USB 转串口线（PL2303 芯片）。驱动安装参考 [ubuntu 安装 USB 转串口驱动](#)

(2) 第二步，安装 `mimicom` 串口工具，并能够正确识别插上的 USB 转串口。具

体自行查阅资料。成功连接串口界面如图3-2:

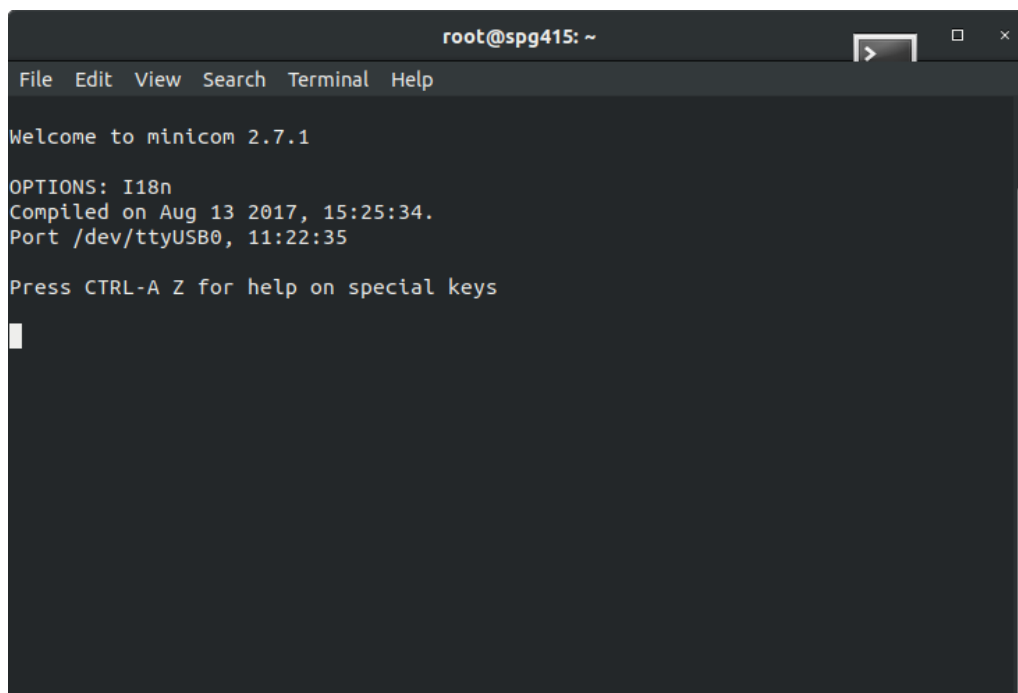


图 3-2 minicom 初始界面

(3) 第三步，将开发板左下角的开关向下拨到 nor，如图3-3所示。

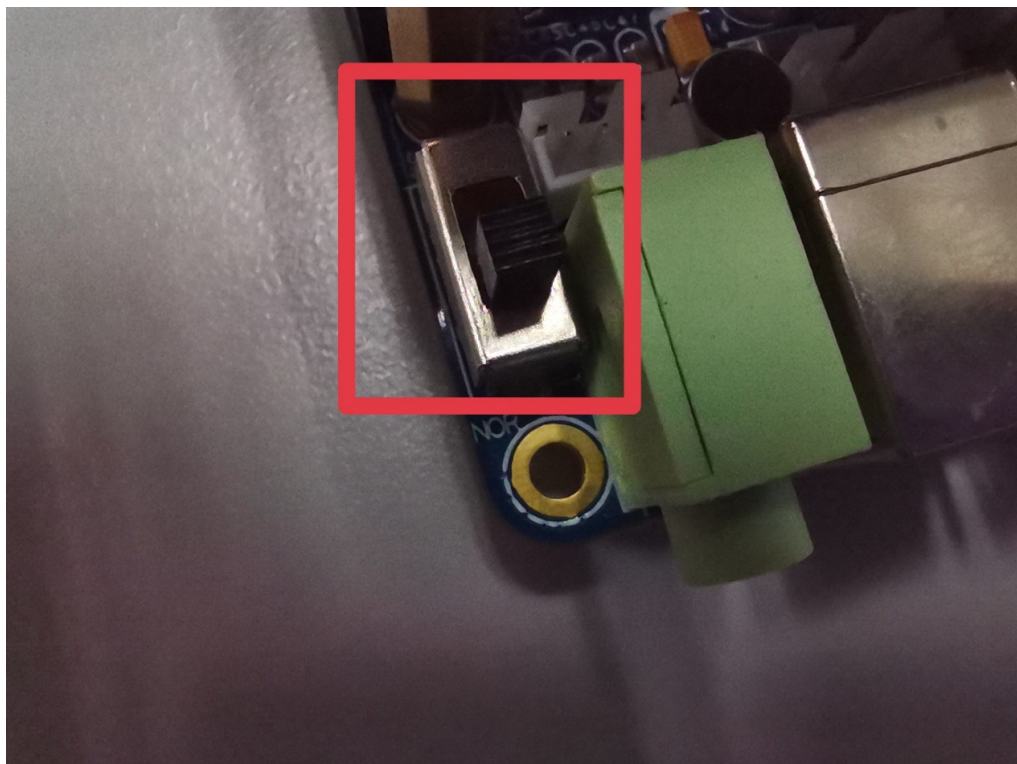
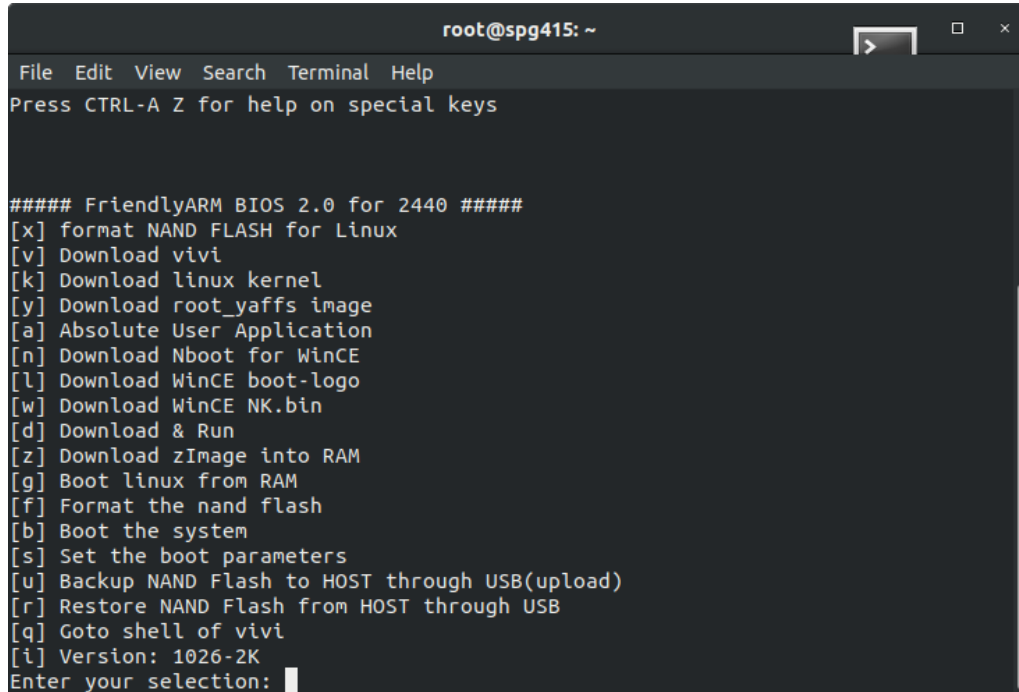


图 3-3 nor 开关

(4) 第四步，编译之前下载的工具目录下的 `dnw.c` 文件。这里我们已经提前编译好了 `dnw`，直接使用即可。`dnw` 就是 USB 下载线的驱动。

(5) 运行 `minicom`，打开 `mini2440` 的电源，你将看到如图3-4所示界面：

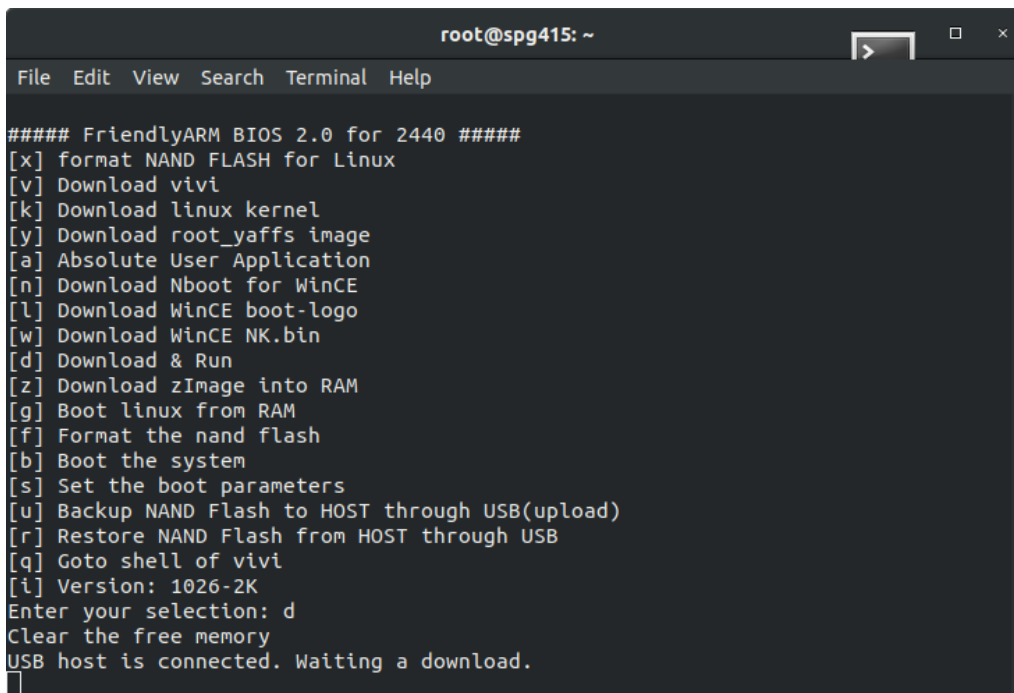


```
root@spg415: ~
File Edit View Search Terminal Help
Press CTRL-A Z for help on special keys

##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 1026-2K
Enter your selection: 
```

图 3-4 supervivi 界面

输入 `d`，看到如图3-5所示界面：

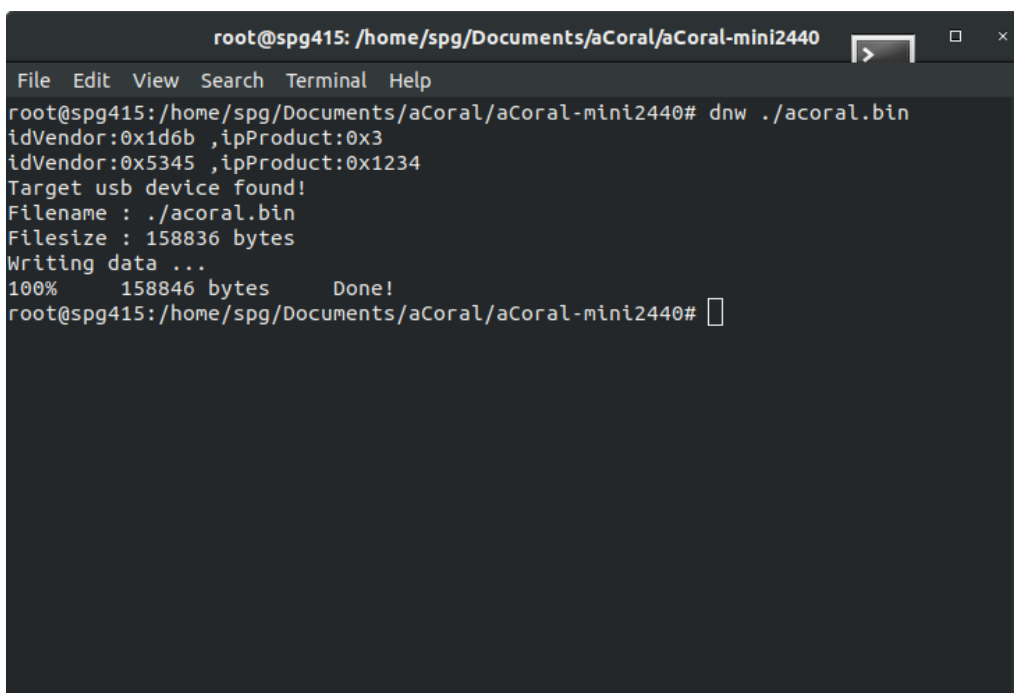


```
root@spg415: ~
File Edit View Search Terminal Help

##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 1026-2K
Enter your selection: d
Clear the free memory
USB host is connected. Waiting a download.
█
```

图 3-5 等待 USB 下载

运行 dnw，下载 acoral.bin，如图3-6：



```
root@spg415: /home/spg/Documents/aCoral/aCoral-mini2440
File Edit View Search Terminal Help

root@spg415:/home/spg/Documents/aCoral/aCoral-mini2440# dnw ./acoral.bin
idVendor:0x1d6b ,ipProduct:0x3
idVendor:0x5345 ,ipProduct:0x1234
Target usb device found!
Filename : ./acoral.bin
Filesize : 158836 bytes
Writing data ...
100% 158846 bytes Done!
root@spg415:/home/spg/Documents/aCoral/aCoral-mini2440# █
```

图 3-6 dnw 下载

如果一切顺利，你将在 minicom 看到如图3-7所示的界面：

```

root@spg415: ~
File Edit View Search Terminal Help
USB host is connected. Waiting a download.

Now, Downloading [ADDRESS:30000000h,TOTAL:158846]
RECEIVED FILE SIZE: 158846 (155KB/S, 1S)
hello spgin init spg30000000, size = 158836 bytes
mcoral_net_lwip_init
DM9000 init spggogogo***** send DATA*****
len: 52the packed len: 52 <--->
ff ff ff ff ff ff
0 11 22 33 44 55
8 0 45 0 0 26
0 0 0 0 ff 11
f9 6d c0 a8 1 b1
ff ff ff ff 10 0
cf b1 0 12 35 18
73 70 67 67 6f 67
6f 67 6f 0
<*<*>*>*>

+++++
done@
udp send to router done!
aCoral:>

```

图 3-7 aCoral shell

大家可以在中输入 help 命令，看看 aCoral 现在支持的指令，如图3-8所示：

```

root@spg415: ~
File Edit View Search Terminal Help
len: 52the packed len: 52 <--->
ff ff ff ff ff ff
0 11 22 33 44 55
8 0 45 0 0 26
0 0 0 0 ff 11
f9 6d c0 a8 1 b1
ff ff ff ff 10 0
cf b1 0 12 35 18
73 70 67 67 6f 67
6f 67 6f 0
<*<*>*>*>

+++++
done@
udp send to router done!

aCoral:>help
memscan View the first Level Memory Managment Info
dt View all thread info
spg Easter egg
exit Exit Shell
help View all Shell Command info
aCoral:>

```

图 3-8 help 命令

至此，aCoral 的下载完成，之后如果又重新编译了新的镜像，都可以按照这个步骤来。需要注意的是，这种使用 bootloader 直接往 sdram 下载的镜像，关机之后

没了，需要重新下载。



## 第四章 相关资料

根目录下的 `Appendix` 文件夹内放有在学习 `aCoral` 过程中会用到的一些第三方文档。