DOI:10.16468/j.cnki.issn1004-0366.2011.04.014

基于 ARM9 的嵌入式 Linux 系统分析与移植

缑新科，滕 永
（兰州理工大学 电气工程与信息工程学院，甘肃 兰州 730050）

摘 要： 结合 S3C2440 处理器和 Mini2440 实验平台，进行了嵌入式 Linux 的移植. 实施了 U-Boot 移植，解决了 U-Boot 中关于 Nand-Flash 启动问题. 分析了 Linux 的内核结构，从而实现了内核移植. 测试结果证明该方法是可行的.

关键词： Linux；U-Boot；ARM9；嵌入式系统
中图分类号： TP316 **文献标志码：** A **文章编号：**1004-0366(2011)04-0106-04

Analysis and Porting of Embedded Linux System Based on ARM9

GOU Xin-ke, TENG Yong
(*College of Electrical and Information Engineering, Lanzhou University of Science and Technology, Lanzhou* 730050, *China*)

Abstract: Based on S3C2440 processor and Mini2440 platform, the porting of the embedded Linux system was carried out successfully. First, based on solving the problem of Nand-Flash, the U-Boot was ported to the platform. Second, the kernel of the Linux was analyzed and ported too. The test results suggest that this method is feasible.

Key words: Linux; U-Boot; ARM9; embedded system

嵌入式系统和 PC 系统启动的方式类似，都需要有一定的引导程序. 在 PC 机启动的时候，首先运行 BIOS(Basic Input/Output System). 这个系统可以把系统从硬件启动过渡到软件管理中，为下一步从硬盘中将操作系统调用至内存运行做好准备. 而嵌入式系统的这种类似于 BIOS 的系统，我们称之为 Boot-loader[1]，Boot-loader 与系统的硬件息息相关. 而在嵌入式系统中，不同结构的 CPU 其对应的 Boot-loader 也都是不同的，即便开发板所选用的处理器是同一型号，仅仅是外设的不同，也需要进行 Boot-loader 的移植工作. 因此想要在嵌入式世界里建立一个通用的 Boot-loader 几乎是不可能的.

Linux 是源代码开放的，支持多用户、多进程、多线程、实时性较好的功能强大的操作系统. 它可以运行在 x86，680x0，PowerPC，MIPS，ARM 等平台上，已经是目前运行硬件平台最多的操作系统. 我们在此基础上提出了将功能强大的 U-Boot 和 Linux 内核结合移植到特定目标板上的思路和方法.

1 U-Boot 分析及启动过程

U-Boot(Das u-Boot)其含义为 Universal Boot-Loader，是由德国 DENX 软件工程中心开发和维护的针对嵌入式 CPU 的 Boot-loader，是遵循 GPL 条款的开放源码项目[2].

1.1 U-Boot 启动分析

U-Boot 的启动分为两个阶段. 在第一个阶段主要完成以下工作：

(1) 基本的硬件初始化；

(2) 为下一阶段准备 RAM 空间；

(3) 将第二阶段所需数据转移至 RAM 中；

(4) 设置堆栈指针和存取方式，准备好第二阶段需 C 语言环境.

第二阶段的主要工作：

收稿日期：2011-04-14

(1) 从汇编语言跳转到 main 入口函数;

(2) 初始化本阶段要使用的硬件设备;

(3) 检测系统内存映射;

(3) 加载内核映像和文件系统.

1.2 U-Boot 结构分析

为了实现 U-Boot 的移植,首先必须对 U-Boot 中的文件结构有一定的了解.结合本平台的需要,我们主要在 ARM 环境下对主要部分作一定分析. U-Boot结构如图 1 所示.

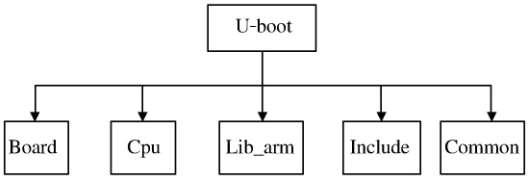

图 1　U-Boot 结构

图 1 中 Board 目录为依赖于具体的平台;CPU 目录为 U-Boot 所支持的 CPU;Lib_arm 目录为针对 ARM 结构体系的通用文件;Include 目录为头文件和开发板配置文件;Common 目录主要是用于实现各种命令的 C 文件.

2 U-Boot 移植过程

2.1 系统结构与文件配置

实验平台是由三星 S3C2440 微控制器、128 M NandFlash、64 M SDRAM、串口、以太网口、JTAG 口等组成,其功能如图 2 所示.

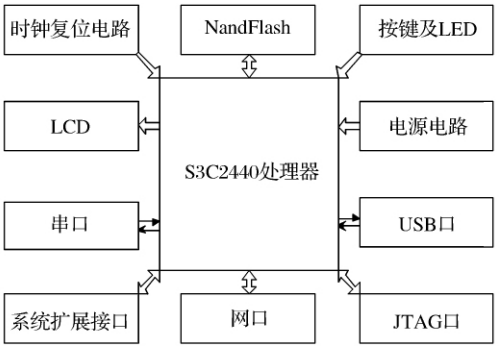

图 2　开发板的组成

为了缩短开发周期,通常都是选用与自己开发板硬件相似,且已经获得 U-Boot 支持的平台为模板来进行移植. 选用 SMDK2410,因此移植所要做的工作主要是针对 S3C2410 和 S3C2440、SMDK2410 和 S3C2440board 的外设不同做相应的修改,并添加新的功能.

我们在执行编译之前,首先要对 Makefile 文件添加

S3C2440_config:unconfig@. mkconfig $(@:_config=)arm arm920t S3C2440.

配置完成后,在 board 中建立 S3C2440board 目录,并复制 smdk2410 目录中的内容,之后执行 make smdk2410_config.

其次在 include/configs/中建立配置头文件,然后指定交叉编译工具的路径以及测试编译.

以上的工作是 U-Boot 的移植前序步骤,不过这里存在一个问题:代码搬运. 本平台是基于 Nand-Flash 启动,但 U-Boot 并不支持这一启动方式,这就需要我们对这部分进行一定的修改和完善.

2.2 NandFlash 中的代码搬移

(1) 前 4 K 的问题　一旦 S3C2440 配置成从 NandFlash 启动,前 4 K 数据自动复制到 4 K 的内存 RAM 中,并将 0x00000000 作为内部 RAM 的起始地址,CPU 从 0x00000000 的内存位置开始运行汇编程序. 程序员要实现的就是如何在 4 K 大小的空间中添加最核心的系统启动程序.

(2) 启动程序的流程　在启动程序的前 4 K 空间,必须完成两个任务:一是完成 S3C2440 的核心配置;二是如何把启动程序(UBOOT)的其余部分拷贝到 RAM 中.

(3) NandFlash 详细设置　首先在 include/configs/smdk2410. h 中加入 CONFIG_S3C2440_NAND_BOOT,如下:#define CONFIG_S3C2440_NAND_BOOT 1 支持从 NandFlash 中启动. 之后在 start. S 中添加操作 NandFlash 控制器的代码,并且在另一个 C 语言文件中实现 NandFlash 的读函数 nand_read(). 要注意的是在 Makefile 中添加对这个文件的编译. 这样就实现了 U-Boot 重定位功能,即将 U-Boot 从 NandFlash 拷贝到 SDRAM 中. 程序流程如图 3 所示.

2.3 NandFlash 读写命令问题

在 U-Boot 下主要是通过命令行的方式来实现对 NandFlash 的操作. 用到的主要数据结构有:struct nand_flash_dev、struct nand_chip. 前者包括设备的信息,诸如芯片型号、存储容量、设备 ID、I/O 总线宽度等信息;后者是 NandFlash 工作时所支持的信息. U-Boot 支持 NandFlash 命令移植主要是有如下的步骤:设置配置选项、加入自己的 NandFlash 芯片型号、编写自己的 NandFlash 初始化函数. 以上就完成了 U-Boot 对 NandFlash 的支持,使用交叉工具链编译完善之后的 U-Boot,生成 uboot. bin

的二进制文件. 使用 Jtag 将编译好的 U-Boot 烧写到开发板中去.

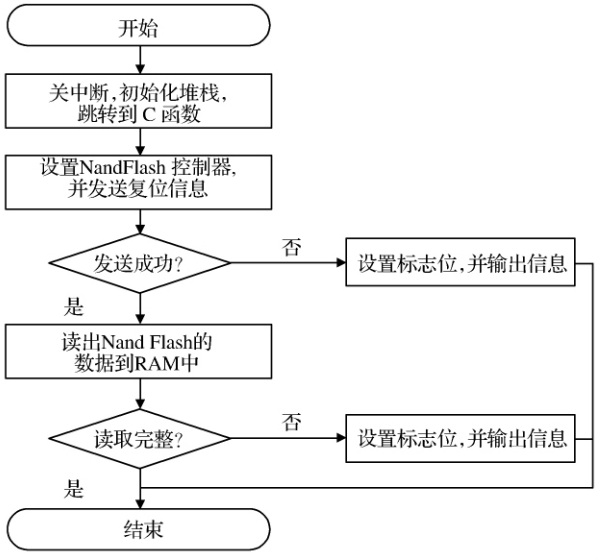

图 3 NandFlash 流程

2.4 移植结果

U-Boot 中的 Makefile 文件体现了 U-Boot 内部代码之间的依赖关系. 而要生成相应映像文件,只要通过使用 make 命令就可以实现. 之后使用交叉开发工具链 arm-gcc-linux 就可以生成二进制的文件. 成功移植后的串口输出 U-Boot 信息如图 4 所示.

```
U-Boot1.1.6 (Jan 6 2011 - 16:16:50)
DRAM:  64MB
Flash:  2MB
NAND:  128MB
*** Warning - bad CRC, using default environment
In    :  serial
Out   :  serial
Err   :  serial
Mini2440@nand info
```

图 4 U-Boot 烧写结果

3 Linux 内核移植过程

Linux 操作系统因其开放源代码、易于开发、功能强大、稳定、成本低等优势迅速跻身于主流嵌入式开发平台. 对于 Linux 在嵌入式系统中的应用,都是根据各种系统的多样性,对内核进行一定的裁剪和修改,而不是生搬硬套.

3.1 Linux 结构分析

Linux 内核主要由 5 个子系统组成:进程调度、内存管理、虚拟文件系统、网络接口、进程间通信.

Linux 内核非常庞大,一个功能齐全,包罗万象的内核有数百兆之多,其整体分布如图 5 所示.

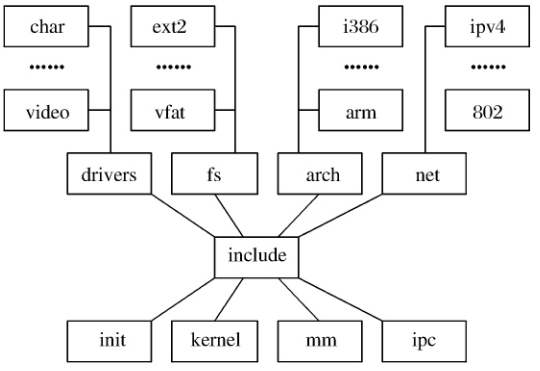

图 5 Linux 内核组织结构

3.2 移植过程

(1)下载 Linux 内核并修改 Makefile 文件 http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.29.5.tar.bz2 下载 linux2.6.29.5 内核,并修改内核目录树根下的 Makefile,找到 ARCH 和 CROSS_COMPILE,修改 ARCH ?= arm CROSS_COMPILE ?= arm-linux-gcc,然后设置路径环境变量,使其可以找到对应的交叉编译工具链.

(2) 指明分区信息 ① 建立 NandFlash 分区表:目标板计划分 4 个区,分别存放 U-Boot,内核,文件系统以及用户应用系统空间;② 加入 NandFlash 分区;③ 建立 NandFlash 芯片支持;④ 加入支持 NandFlash 的驱动.

(3) 指定启动时初始化 当内核启动时,我们需要依据之前对分区的设置来进行初始化配置,主要修改 arch/arm/machS3C2410/machsmdk2410.c 文件.

(4) 禁止 Flash ECC 校验 由于 U-Boot 所使用的是由软件 ECC 算法产生 ECC 校验码,这与内核校验的 ECC 码不一样,所以我们在这里选择禁止内核 ECC 校验. 在 driver/mtd/nand/S3C2410.c 这个文件中,我们找到 S3C2410_nand_init_chip()函数,在该函数体最后一行加上一条语句:chip>ecc-mode=NAND_ECC_NONE.

(5) 配置并编译内核 对于内核的配置,我们常用的方法就是在内核目录下执行:make menuconfig 命令. 图形化内核配置窗口见图 6. 在这里有选择的选取内核的部分功能,以达到缩小内核的体积的目的.

在对内核配置完成后,可以通过编译工具 arm-gcc-linux 来编译刚刚保存的内核. 这可以生成内核映像文件. 通过之前已经烧写好的 U-Boot,可以将内核也烧写到 NandFlash 中去.

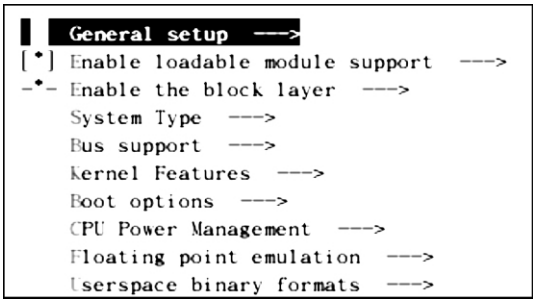

图 6　内核配置窗口

4　结语

Boot-loader 是系统上电后运行的第一段代码,主要完成一系列初始化工作. U-Boot 是比较通用的开源 Boot-loader. 它功能强大,支持多种 CPU 架构,多种操作系统内核. 这使得 U-Boot 成为主流的嵌入式 Boot-loader,同时 Linux 操作系统因源码开放、易于开发、功能强大、稳定、成本低等优势迅速称为嵌入式系统中的主流操作系统. 将这二者相结合,对嵌入式系统来说意义重大. 在简要地介绍了 Boot-loader 和 Linux 操作系统的基础上,比较详细地分析了 U-Boot 和 Linux 的性能特点.

在具体实践中将 U-Boot 移植到基于 ARM920t 的处理器 S3C2440 上并给出了其大致过程,给出了实现 U-Boot 自动识别启动 Flash 是 NandFlash 还是从 NorFlash 的原理,同时也给出了编写相关代码的思路. 并在此基础上根据硬件的特点,给出了 Linux 操作系统移植的详细思路和步骤.

参考文献:

[1] Sonia Thakur, James M Conrad. An Embedded Linux Based Navigation System for an Autonomous Underwater Vehicle [J]. IEEE, 2007: 56-58.

[2] 田泽. 嵌入式系统开发与应用[M]. 北京:北京航空航天大学出版社, 2005.

[3] 何景波. 基于 Linux 的嵌入式应用系统技术研究[D]. 太原:中北大学, 2009.

[4] Christopher Hallinan. Embedded Linux Primer[M]. 影印版. 北京:人民邮电出版社, 2008.

[5] 颜华. 基于 ARM 的嵌入式 Linux 操作系统研究与移植[D]. 北京:北京工业大学, 2007.

[6] Karim Yaghmour, Jon Master. Building Embedded Linux Systems[M]. 2rd. O'Reilly, 2008.

[7] Michael Barr, Anthony Massa. Programming Embe dded Systems[M]. 影印版. 南京:东南大学出版社, 2007.

[8] ADI. ADSP-BF561 Blackfin Processor Hardware Reference [Z]. 2003.

[9] DAVICOM Semiconductor Inc. DM9000 A Ethern et Controller with General Processor Interface Data Sheet[M]. 2006.

[10] 三恒心科技有限公司编著. ARM9 原理与应用设计[M]. 北京:电子工业出版社, 2008.

作者简介:

缑新科 (1966-),男,甘肃省天水人,现任兰州理工大学电气工程与信息工程学院教授,硕士研究生导师. 主要研究方向为智能结构及其动力学系统控制.