

深度实践嵌入式Linux系统移植



[深度实践嵌入式Linux系统移植_下载链接1](#)

著者:范展源

出版者:机械工业出版社

出版时间:2015-5

装帧:平装

isbn:9787111497912

随着物联网时代的到来，市场对各类智能设备的需求也日渐高涨。智能设备的核心技术是集成电路芯片和嵌入操作系统，而嵌入式操作系统更可以称为是智能设备的灵魂。多年来，Linux系统因为其开源免费、安全稳定、社区支持丰富和移植裁剪方便等特点一直备受全球各大设备厂商的青睐，当仁不让地成为众多嵌入式操作系统中最耀眼的明星。在智能手机大行其道的今天，Android系统牢牢占据着80%以上的市场份额，而在Android系统华丽的外衣之下，依然是Linux系统强有力的支撑。

智能设备的发展并没有止步在智能手机，而是飞速朝着智能可穿戴设备、智能家居和车联网等方向扩张。但要想把Linux系统移植到这些外设丰富并且处理器架构各异设备中，对于初级工程师而言也并非易事，因为这不仅要求工程师熟悉c、汇编等编程语言，了解基本的硬件操作方法和协议规范，还要求工程师对Linux的内核架构、编译系统、调试方法以及各个子系统的源码结构有所理解。为了让读者能够顺利具备Linux系统移植的能力，本书被设计为一站式学习教程，即：

- 涉及Linux系统各个层面的移植，包括启动加载程序、Linux内核、Linux应用程序等；
- 提供深入的理论讲解和完整的源码剖析，同事也分析了启动加载程序和Linux内核的编译系统；
- 分别提供对使用ARM9／S3C2440和ARM11／S3C6410两款处理器的开发板移植过程的详细实录，以最为人性化的方式让读者理解整个移植过程中代码和系统功能的变迁。

作者介绍:

为什么要写这本书

近年来，以嵌入式技术为基础的物联网产业正如火如荼地发展，就业需求和薪资待遇年年看涨。许多学校和培训机构纷纷开设了嵌入式相关的课程，但这些课程的实践环节往往存在以下几个方面的问题：

过于偏重理论，缺乏对实践环节的指导。

有实践环节，但内容过于简单，流于表面。

有实践环节，但移植实践部分讨论的内容不完整。

由于存在上述的问题，通过这些课程培养出来的学生往往满足不了企业的真正需求。因此，相关学员也很难圆高薪就业的梦想。

本书力求为上述三个问题寻求解决方案：第一，在注重理论知识的前提下，坚持以实践环节为本书内容的主体，从实践中提炼理论知识，以源码分析作为移植实践的依据；第二，绝不浅尝辄止的实践，绝不停留于只求在某个开发板上做出某个简单的效果，而是从对多个开发板的移植实践中提炼出更一般性的方法；第三，力求做到功能完善的系统移植。虽然嵌入式系统往往根据需求对功能进行裁剪，但为了打造一本面向广大嵌入式工程师和爱好者的书，笔者还是决定向读者介绍更多功能的移植方法，以让本书可以作为一本工具书常备、常用。

读者对象

这里可以根据软件需求来划分本书的读者：

Linux系统爱好者

嵌入式u-boot移植工程师

嵌入式Linux移植工程师

Linux驱动开发工程师

嵌入式应用开发工程师

高校相关课程的教师与学生

参加嵌入式职业培训的人员

如何阅读本书

本书共23章，分为四篇。

第一篇为绪论篇（第1章）。本篇只有一章内容，从整体上简单地介绍了嵌入式系统架构，并探讨了系统移植环境的搭建方法。本章中的所有内容均以介绍和指导实践为主，读者仅需了解相关的概念，之后自会在移植实践过程中有深刻体会。

第二篇为u-boot移植篇（第2~5章）。从绪论篇中得知，嵌入式系统的移植首先需要完成的工作是启动加载程序的移植，由于本书是以实践为目的，因此选择了嵌入式系统最为常用的u-boot作为移植的依据。

本篇首先介绍u-boot工程与编译系统以及u-boot启动流程等基础知识，并深入源码进行详细分析。读者在掌握这些基础知识后，可以参照使用ARM9/S3C2440芯片的开发板和使用ARM11/S3C6410芯片的开发板这两个实例进行系统的u-boot移植实战。由于力求深入与完整，移植实战涵盖了从最初的启动代码到各种常见驱动的移植，并在最后整合所有的移植功能，完成一个功能强大的启动菜单。所有的移植代码和相关资源均可在本书配套资源文件中找到。

第三篇为Linux内核移植篇（第6~18章）。本篇是本书的核心内容，与第二篇类似，在正式开始内核移植前，首先介绍Linux内核工程及其编译系统、Linux内核相关的调试技术，以及Linux内核的启动流程，为内核移植的讲解打下基础。然后以使用ARM9/S3C2440芯片的开发板和使用ARM11/S3C6410芯片的开发板为例，进行彻底的系统移植。

第四篇为应用程序移植篇（第19~23章）。在完成操作系统底层移植后，为了能在开发板上实现具体的应用，安排了本篇内容。本篇将从嵌入式图形界面移植、嵌入式多媒体移植、嵌入式数据库移植、嵌入式Web服务器移植和嵌入式JVM移植等方面介绍嵌入式应用程序的实践过程。

勘误和支持

由于作者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为此，特意创建一个在线支持与应急方案的二级站点<http://book.blendern.cn.org>。你可以将书中的错误发布在Bug勘误表页面中，同时如果你遇到任何问题，也可以访问Q&A页面，我将尽量在线上为你提供最满意的解答。书中的全部资源文件除可以从华章网站（www.hzbook.com）下载外，还可以从我创建的这个站点下载，我也会将相应功能的更新及时更正出来。如果你有更多的宝贵意见，也欢迎发送邮件至邮箱lightfather@gmail.com，期待能够得到你们的真挚反馈。

致谢

首先要感谢伟大的Linux创始人Linus Torvalds，是他开创了一款影响整个嵌入式行业的操作系统。

感谢成都国嵌信息技术有限公司，尤其要感谢国嵌的谢伟老师，是他建议我将自己的嵌入式移植实战经验编写成书，他不仅是一位知识渊博的老师，也是一位实战经验丰富的嵌入式工程师。在编写本书的过程中，得到了谢伟老师很多宝贵的建议。

感谢全国众多的国嵌学员，我们曾经常在一起讨论本书应该以什么形式编写，正是这些源自潜在读者的宝贵建议，促成了本书以移植实践为目的的写作动机。

感谢机械工业出版社华章公司的编辑姜影老师，在这一年多的时间中始终支持我的写作，她的鼓励和帮助引导我能顺利完成全部书稿。

谨以此书献给众多热爱嵌入式Linux的朋友们！

目录: 前言

绪论篇

第1章嵌入式系统架构与移植环境搭建2

1.1 嵌入式系统硬件架构2

1.1.1 微处理器3

1.1.2 总线4

1.1.3 存储器5

1.2 嵌入式系统软件架构6

1.3 嵌入式Linux移植环境搭建7

1.3.1 Ubuntu开发平台7

1.3.2 搭建交叉编译环境8

1.3.3 获取内核9

1.3.4 获取启动加载器9

1.3.5 配置必要服务9

1.3.6 PuTTY的安装和配置12

1.4 本章小结12

u-boot移植篇

第2章u-boot工程与编译系统14

2.1 u-boot介绍14

2.1.1 u-boot工程简介15

2.1.2 u-boot源码结构15

2.1.3 u-boot的配置编译16

2.2 u-boot常用命令与测试18

2.2.1 获取帮助18

2.2.2 环境变量相关命令19

2.2.3 网络命令20

2.2.4 Nand Flash操作命令21

2.2.5 内存/寄存器相关命令22

2.2.6 系统引导命令24

2.3 u-boot编译过程分析25

2.3.1 主机构建环境配置过程25

2.3.2 目标机相关配置过程27

2.3.3 make命令执行过程31

2.4 本章小结40

第3章u-boot启动流程分析41

3.1 u-boot启动第一阶段流程41

3.1.1 设置异常向量42

3.1.2 CPU进入SVC模式43

3.1.3 设置控制寄存器地址43

3.1.4 关闭看门狗43

3.1.5 屏蔽中断43

3.1.6 设置MPLLCON、UPLLCON和CLKDIVN44

3.1.7 关闭MMU和cache45

3.1.8 初始化存储控制器46

3.1.9 复制u-boot第二阶段代码到RAM47

3.1.10 设置栈48

3.1.11 清除BSS段48

3.1.12 跳转到第二阶段代码入口48

3.2 u-boot启动第二阶段代码分析49

3.2.1 gd_t结构体49

3.2.2 bd_t结构体50

3.2.3 init_sequence数组	50
3.2.4 start_armboot()顺序分析	51
3.2.5 main_loop函数分析	52
3.3 本章小结	56
第4章ARM9/S3C2440 u-boot移植实战	57
4.1 移植准备工作	57
4.1.1 开发环境介绍	57
4.1.2 删减u-boot文件	58
4.1.3 建立My2440配置	59
4.1.4 配置和编译u-boot	60
4.2 u-boot芯片级移植	61
4.2.1 启动代码结构优化	61
4.2.2 系统时钟移植	65
4.2.3 存储器控制器设置	68
4.2.4 在u-boot工程中全面支持CONFIG_S3C2440宏配置	69
4.3 u-boot调试方法探索	70
4.3.1 通过LED指示运行状态	70
4.3.2 在第一阶段的代码中添加打印调试方法	72
4.3.3 在内存中加载和运行u-boot	78
4.4 Nor Flash驱动移植	78
4.4.1 Nor Flash的工作模式	78
4.4.2 Nor Flash的存储结构	79
4.4.3 Nor Flash的硬件连接	79
4.4.4 Nor Flash的操作方法	80
4.4.5 Nor Flash驱动分析	83
4.4.6 Nor Flash驱动移植	86
4.4.7 Nor Flash命令测试	87
4.4.8 完善u-boot的命令行功能	88
4.5 DM9000驱动移植	89
4.5.1 DM9000网卡端口访问	90
4.5.2 DM9000网卡时序分析	90
4.5.3 DM9000网卡驱动分析	93
4.5.4 DM9000网卡驱动移植	100
4.5.5 网卡驱动测试	103
4.5.6 通过TFTP下载程序到内存运行	103
4.6 u-boot启动内核	105
4.6.1 ARM架构的Linux内核启动	105
4.6.2 内核标记列表	106
4.6.3 u-boot启动命令分析1—go命令	109
4.6.4 u-boot启动命令分析2—bootm命令	110
4.6.5 u-boot启动命令的配置与移植	115
4.7 Nand Flash与驱动移植	118
4.7.1 Nand Flash启动原理	118
4.7.2 Nand Flash硬件特性	119
4.7.3 Linux MTD子系统	121
4.7.4 Nand Flash初始化流程分析	123
4.7.5 Nand Flash命令实现分析	125
4.7.6 页读取操作详解	127
4.7.7 Nor Flash和Nand Flash启动自动判断	132
4.7.8 Nand Flash拷贝代码实现	132
4.7.9 Nand Flash板级驱动移植	136
4.7.10 Nand Flash命令测试	140
4.8 YAFFS文件系统移植	142
4.8.1 YAFFS文件系统	142

4.8.2 YAFFS在Nand Flash上的存储结构	142
4.8.3 YAFFS在内存中的组织结构	143
4.8.4 在u-boot中添加对烧写YAFFS镜像的支持	144
4.8.5 YAFFS文件系统镜像制作	150
4.8.6 YAFFS的烧写与测试	152
4.9 UBIFS文件系统移植	153
4.9.1 UBI层	153
4.9.2 UBIFS介绍	155
4.9.3 在u-boot中添加对UBIFS的支持	156
4.9.4 制作UBIFS文件系统镜像	157
4.9.5 UBIFS的烧写与测试	158
4.10 SD卡驱动移植	162
4.10.1 MMC/SD/SDIO介绍	162
4.10.2 SD/MMC协议	162
4.10.3 S3C2440 SDI控制器操作	166
4.10.4 SD卡驱动分析	168
4.10.5 SD卡驱动移植	173
4.11 USB驱动移植	176
4.11.1 USB概述	176
4.11.2 USB系统架构	177
4.11.3 USB的通信方法	180
4.11.4 USB的描述符	184
4.11.5 USB设备请求	188
4.11.6 USB设备枚举	191
4.11.7 S3C2440 USB设备控制器	195
4.11.8 u-boot USB设备控制器驱动分析	196
4.11.9 USB设备驱动移植	206
4.11.10 USB驱动移植	210
4.11.11 USB功能测试	214
4.12 u-boot一键式菜单实现	215
4.12.1 一键式菜单需求分析	215
4.12.2 一键式菜单测试步骤	216
4.12.3 一键式菜单源码分析	220
4.13 本章小结	224
第5章ARM11/S3C6410 u-boot移植实战	225
5.1 移植准备工作	225
5.1.1 开发环境	225
5.1.2 删减u-boot文件	226
5.1.3 建立My6410配置	227
5.1.4 配置和编译u-boot	228
5.2 u-boot芯片级移植	229
5.2.1 修改第一阶段启动代码start.S	229
5.2.2 板级底层初始化文件lowlevel_init.S移植	233
5.2.3 时钟初始化函数移植	239
5.2.4 内存初始化函数实现	243
5.2.5 Nand Flash复制代码实现	246
5.2.6 SD卡复制代码实现	250
5.2.7 底层调试方法探索	254
5.2.8 完善My6410的板级配置	258
5.2.9 烧写与测试	265
5.3 DM9000驱动移植	267
5.3.1 DM9000网卡端口访问	267
5.3.2 DM9000网卡时序分析	268
5.3.3 DM9000网卡驱动移植	270

- 5.3.4 网卡驱动测试274
- 5.4 u-boot启动内核274
 - 5.4.1 u-boot启动命令的配置与移植275
 - 5.4.2 修改go命令以支持zImage格式镜像的启动277
- 5.5 Nand Flash驱动移植278
 - 5.5.1 将u-boot下载到内存中运行278
 - 5.5.2 Nand Flash驱动移植279
 - 5.5.3 Nand Flash命令测试289
- 5.6 YAFFS文件系统移植291
 - 5.6.1 在u-boot中添加对烧写YAFFS镜像的支持291
 - 5.6.2 YAFFS文件系统镜像制作298
 - 5.6.3 YAFFS的烧写与测试299
- 5.7 UBIFS文件系统移植300
 - 5.7.1 在u-boot中添加对UBIFS的支持300
 - 5.7.2 UBIFS文件系统镜像制作302
 - 5.7.3 UBIFS的烧写与测试303
- 5.8 SD卡驱动移植307
 - 5.8.1 S3C6410 主机控制器操作307
 - 5.8.2 S3C6410 主机控制器操作序列308
 - 5.8.3 SD卡驱动分析310
 - 5.8.4 SD卡驱动移植317
 - 5.8.5 通过SD卡更新系统319
- 5.9 USB驱动移植321
 - 5.9.1 S3C6410 USB2.0高速OTG321
 - 5.9.2 u-boot USB设备控制器驱动分析323
 - 5.9.3 USB设备驱动移植332
 - 5.9.4 USB功能测试335
- 5.10 本章小结336
- Linux内核移植篇
- 第6章Linux内核工程与编译系统338
 - 6.1Linux内核架构338
 - 6.1.1内核体系结构338
 - 6.1.2内核组件339
 - 6.1.3内核目录结构340
 - 6.2 Linux内核的配置与编译341
 - 6.2.1 配置内核341
 - 6.2.2编译内核344
 - 6.3Linux内核构建系统345
 - 6.3.1内核配置过程345
 - 6.3.2扩展内核代码347
 - 6.3.3内核中的Makefile348
 - 6.3.4内核中的Kconfig349
 - 6.4内核调试技术352
 - 6.4.1调试准备352
 - 6.4.2内核调试配置选项352
 - 6.4.3源码级别的调试接口353
 - 6.4.4使用printk()打印调试信息355
 - 6.4.5使用strace跟踪系统调用357
 - 6.4.6使用OOPS调试系统故障358
 - 6.5本章小结360
- 第7章Linux内核启动流程分析361
 - 7.1内核镜像生成361
 - 7.2内核启动流程1——汇编部分362
 - 7.2.1内核启动代码入口362

7.2.2深入源码分析	363
7.2.3汇编启动代码分析总结	378
7.3内核启动流程2—C语言部分	378
7.3.1start_kernel()函数	379
7.3.2rest_init()函数	388
7.3.3 kernel_init()函数	390
7.3.4init_post()函数	391
7.4内核启动流程3—Busybox的init进程	393
7.4.1init进程启动流程	393
7.4.2添加初始化活动	394
7.4.3执行初始化活动	395
7.5本章小结	396
第8章Linux移植准备及最小系统构建	397
8.1移植准备工作	397
8.1.1 开发环境	397
8.1.2删减Linux文件	398
8.1.3建立My2440配置	400
8.1.4 建立My6410配置	403
8.1.5编译测试	406
8.2 最小系统搭建	409
8.2.1 嵌入式根文件系统制作	409
8.2.2安装initramfs根文件系统	412
8.3本章小结	414
第9章Linux网卡驱动移植	415
9.1Linux网络子系统	415
9.2 核心数据结构	416
9.2.1 net_device结构	416
9.2.2 sk_buff结构	419
9.3 DM9000网卡驱动分析	421
9.3.1 board_info结构	422
9.3.2 dm9000_probe()函数	423
9.3.3 dm9000_open()函数	427
9.3.4 dm9000_start_xmit()函数	427
9.3.5 数据包接收函数	428
9.3.6 数据包发送函数	429
9.3.7 中断处理函数	431
9.4My2440网卡驱动移植	432
9.4.1 添加DM9000的平台设备	432
9.4.2 在内核配置添加对DM9000的支持	434
9.5 My6410网卡驱动移植	434
9.5.1 添加DM9000的平台设备	434
9.5.2 在内核配置中添加对网络子系统的支持	435
9.5.3 在内核配置添加对DM9000的支持	436
9.6安装NFS根文件系统	436
9.6.1在内核配置添加对NFS的支持	436
9.6.2挂载NFS根文件系统	437
9.7 制作基于共享库的根文件系统	437
9.8本章小结	439
第10章Linux混杂设备驱动	440
10.1 My2440 RTC驱动移植	440
10.2My6410 RTC驱动移植	441
10.2.1修改RTC驱动rtc-s3c.c	441
10.2.2完善对6410 RTC驱动的平台支持	445
10.2.3 在机器配置文件中添加RTC设备	448

10.2.4 在内核中配置RTC448	
10.3 RTC驱动测试449	
10.4为My2440添加ADC和按键驱动451	
10.4.1按键驱动分析451	
10.4.2 在内核中添加ADC和按键驱动454	
10.5 为My6410添加ADC驱动457	
10.6 本章小结458	
第11章Linux I2C驱动移植459	
11.1 I2C协议概述459	
11.1.1 I2C总线物理拓扑结构459	
11.1.2 I2C通信协议460	
11.2Linux I2C子系统框架461	
11.3 I2C驱动中的数据结构及操作462	
11.3.1 i2c_adapter结构462	
11.3.2 i2c_algorithm结构464	
11.3.3 i2c_msg结构464	
11.3.4i2c_driver结构465	
11.3.5 i2c_client结构467	
11.4I2C适配器的设备接口468	
11.4.1 i2cdev_open()函数471	
11.4.2 i2cdev_read()函数471	
11.4.3 i2cdev_ioctl()函数472	
11.5 S3C2440 (6410) I2C适配器驱动的实现473	
11.5.1 S3C2440 I2C platform总线匹配474	
11.5.2 S3C2440 I2C总线驱动描述结构474	
11.5.3 probe方法的实现476	
11.5.4 S3C2440 I2C总线通信方法477	
11.6 S3C2440 (6410) I2C适配器驱动移植480	
11.6.1 添加 I2C平台设备480	
11.6.2 在内核配置中支持I2C驱动481	
11.6.3编写I2C总线驱动测试程序482	
11.7S3C2440 (6410) I2C设备驱动的实现484	
11.7.1 At24系列I2C EEPROM设备驱动的实现484	
11.7.2传统只读EEPROM设备驱动的实现486	
11.8 I2C EEPROM设备驱动移植489	
11.9本章小结490	
第12章Linux SPI驱动移植491	
12.1SPI协议概述491	
12.1.1 SPI总线物理拓扑结构491	
12.1.2 时钟极性和时钟相位492	
12.1.3 SPI的优缺点493	
12.2 Linux SPI子系统493	
12.3 SPI驱动中的数据结构及操作494	
12.3.1 spi_master结构494	
12.3.2 spi_driver结构495	
12.3.3 spi_device结构496	
12.3.4 spi_message结构497	
12.3.5spi_bitbang结构498	
12.4SPI控制器的设备接口500	
12.5 S3C2440 SPI控制器驱动的实现503	
12.5.1 S3C2440 SPI platform总线匹配503	
12.5.2 S3C2440 SPI控制器驱动描述结构504	
12.5.3 probe方法的实现504	
12.5.4 S3C2440 SPI总线通信方法505	

- 12.6S3C6410 SPI控制器驱动的实现507
- 12.6.1 S3C6410 SPI控制器驱动描述结构507
- 12.6.2 probe方法的实现507
- 12.6.3 S3C6410 SPI总线通信方法509
- 12.7S3C2440 SPI 控制器驱动移植510
- 12.7.1在机器配置文件中添加对SPI的支持510
- 12.7.2 扩展Kconfig512
- 12.7.3在内核配置中支持SPI驱动513
- 12.7.4SPI驱动测试513
- 12.8S3C6410 SPI 控制器驱动移植513
- 12.8.1 添加6410的SPI驱动514
- 12.8.2 添加SPI平台设备514
- 12.8.3添加6410 DMA平台代码516
- 12.8.4 添加SPI、DMA相关的时钟资源518
- 12.8.5在机器配置文件中添加对SPI的支持519
- 12.8.6 SPI驱动测试521
- 12.9 S3C2440 (S3C6410) SPI协议驱动移植521
- 12.9.1 AT25系列SPI EEPROM协议驱动的实现521
- 12.9.2SPI EEPROM设备驱动移植523
- 12.10 本章小结524
- 第13章Nand Flash驱动移植525
- 13.1 Linux MTD子系统525
- 13.2 MTD子系统中的数据结构及操作526
- 13.2.1 mtd_info结构526
- 13.2.2mtd_notifier结构528
- 13.2.3 mtd_part/mtd_partitions结构528
- 13.3 MTD块设备实现分析530
- 13.4 Nand Flash驱动的实现534
- 13.4.1 Nand Flash platform总线匹配534
- 13.4.2 S3C2440 Nand Flash控制器驱动描述结构534
- 13.4.3 probe方法的实现536
- 13.4.4 S3C2440 Nand Flash读写方法分析537
- 13.5S3C2440 Nand Flash控制器驱动移植538
- 13.5.1 在机器配置文件中添加对Nand Flash的支持538
- 13.5.2 完善S3C2440的Nand Flash驱动540
- 13.5.3 在内核配置中支持 Nand Flash驱动541
- 13.6S3C6410 Nand Flash控制器驱动移植541
- 13.6.1添加6410 Nand Flash驱动541
- 13.6.2添加Nand Flash平台设备542
- 13.6.3在机器配置文件中添加对Nand Flash的支持544
- 13.6.4 其他修改545
- 13.7 YAFFS2文件系统移植547
- 13.7.1将YAFFS2文件系统移植到Linux内核548
- 13.7.2 利用mtd-utils烧写YAFFS2文件系统镜像548
- 13.8 在内核中配置对UBIFS的支持550
- 13.9 本章小结550
- 第14章SD/MMC卡驱动移植551
- 14.1Linux MMC子系统551
- 14.2 MMC子系统中的数据结构及操作552
- 14.2.1 mmc_host结构552
- 14.2.2 mmc_card结构554
- 14.2.3mmc_driver结构555
- 14.2.4 mmc_request结构556
- 14.3卡探测过程分析558

- 14.4 S3C2440 SD/MMC主控制器驱动的实现561
 - 14.4.1 SD/MMC主控制器驱动platform总线匹配561
 - 14.4.2 S3C2440 SD/MMC主控制器驱动描述结构561
 - 14.4.3 probe方法的实现563
 - 14.4.4 S3C2440 SD/MMC主控制器请求处理分析565
- 14.5 S3C6410 高速MMC控制器驱动的实现568
 - 14.5.1 SDHCI驱动框架569
 - 14.5.2 SDHCI主机卡探测过程569
 - 14.5.3 S3C6410 HSMMC控制器驱动570
 - 14.5.4 S3C6410 HSMMC控制器请求处理572
- 14.6 S3C2440 SD/MMC主机控制器驱动移植572
 - 14.6.1 在机器配置文件中添加对SD/MMC主控制器的支持572
 - 14.6.2 在内核配置中添加对SD/MMC主控制器驱动的支持573
 - 14.6.3 在内核配置中添加对中文字符集和FAT文件系统的支持574
 - 14.6.4 在文件系统中添加对热插拔事件处理的支持574
 - 14.6.5 SD/MMC主控制器驱动测试575
- 14.7 S3C6410 HSMMC控制器驱动移植576
 - 14.7.1 添加6410 Nand Flash驱动576
 - 14.7.2 在机器配置文件中添加对HSMMC的支持577
 - 14.7.3 修改内核SDHCI驱动578
- 14.8 本章小结579
- 第15章 LCD驱动移植580
 - 15.1 LCD硬件原理580
 - 15.1.1 LCD的硬件构成580
 - 15.1.2 TFT屏显示时序581
 - 15.2 Linux帧缓冲子系统582
 - 15.3 帧缓冲子系统中的数据结构及操作583
 - 15.3.1 fb_info结构583
 - 15.3.2 fb_var_screeninfo结构和fb_fix_screeninfo结构587
 - 15.3.3 fb_cmap结构589
 - 15.4 帧缓冲字符设备接口590
 - 15.5 Linux显示启动Logo594
 - 15.6 S3C2440帧缓冲驱动的实现596
 - 15.6.1 S3C2440 LCD控制器硬件描述596
 - 15.6.2 驱动platform总线匹配597
 - 15.6.3 S3C2440 帧缓冲驱动描述结构598
 - 15.6.4 probe方法实现599
 - 15.7 S3C6410 帧缓冲驱动的实现601
 - 15.8 S3C2440 帧缓冲驱动移植601
 - 15.8.1 在板级初始化文件中添加对帧缓冲的支持601
 - 15.8.2 修改Makefile和Kconfig605
 - 15.8.3 在内核配置中添加对帧缓冲驱动的支持607
 - 15.8.4 帧缓冲驱动测试608
 - 15.9 S3C6410 帧缓冲驱动移植608
 - 15.9.1 添加6410 帧缓冲驱动608
 - 15.9.2 在机器配置文件中添加对帧 缓冲的支持609
 - 15.9.3 帧缓冲驱动测试611
 - 15.10 本章小结611
- 第16章 触摸屏驱动移植612
 - 16.1 Linux输入子系统612
 - 16.2 输入子系统中的数据结构及操作613
 - 16.2.1 input_dev结构613
 - 16.2.2 input_handler结构616
 - 16.2.3 input_handle结构617

- 16.3 输入子系统核心层的实现618
- 16.4 通用事件处理驱动622
- 16.5 输入事件报告流程626
- 16.6 S3C2440（6410）触摸屏驱动的分析629
 - 16.6.1 模块初始化函数的实现629
 - 16.6.2 中断处理与事件上报630
- 16.7 S3C2440触摸屏驱动移植与测试632
 - 16.7.1 S3C2440触屏驱动移植632
 - 16.7.2 S3C2440触屏驱动测试633
- 16.8 S3C6410触摸屏驱动移植与测试634
 - 16.8.1 添加6410的触摸屏驱动634
 - 16.8.2 添加触屏平台设备635
 - 16.8.3 在机器配置文件中添加对触屏的支持636
- 16.9 本章小结637
- 第17章声卡驱动移植638
 - 17.1 ALSA体系架构638
 - 17.1.1 ALSA设备文件639
 - 17.1.2 驱动代码的文件结构640
 - 17.2 声卡描述结构snd_card640
 - 17.3 PCM功能子层644
 - 17.3.1 PCM的概念644
 - 17.3.2 PCM设备描述结构snd_pcm645
 - 17.3.3 PCM流与PCM子流646
 - 17.3.4 PCM功能子层逻辑关系小结651
 - 17.3.5 PCM设备文件的建立652
 - 17.3.6 PCM设备文件的访问654
 - 17.4 声卡控制项655
 - 17.4.1 控制项创建655
 - 17.4.2 控制项回调函数657
 - 17.4.3 Control设备建立658
 - 17.5 ASoC声卡驱动架构659
 - 17.6 ASoC架构中的Machine驱动662
 - 17.6.1 创建ASoC声卡平台设备662
 - 17.6.2 ASoC声卡的平台驱动665
 - 17.7 ASoC架构中的Codec驱动666
 - 17.7.1 Codec的DAI和PCM的配置666
 - 17.7.2 Codec的控制IO669
 - 17.7.3 混音器和其他音频控制项671
 - 17.7.4 Codec设备与ASoC声卡注册673
 - 17.8 ASoC架构中的Platform驱动676
 - 17.8.1 CPU DAI驱动676
 - 17.8.2 音频DMA驱动677
 - 17.8.3 创建音频DMA缓冲区678
 - 17.8.4 音频DMA的PCM操作681
 - 17.9 S3C2440+UDA1341声卡驱动配置与测试683
 - 17.9.1 在机器配置文件中添加对声卡的支持683
 - 17.9.2 在内核配置中支持声卡驱动684
 - 17.9.3 应用层alsa-lib移植685
 - 17.9.4 编写ALSA应用程序686
 - 17.9.5 播放和录音测试690
 - 17.10 S3C6410+WM9714声卡驱动移植691
 - 17.10.1 添加6410声卡驱动691
 - 17.10.2 在内核配置中支持声卡驱动692
 - 17.10.3 alsa-utils工具集移植692

- 17.11 本章小结693
- 第18章USB驱动移植694
 - 18.1USB子系统架构694
 - 18.2USB驱动中的描述符结构695
 - 18.3USB主机驱动695
 - 18.3.1主机控制器驱动695
 - 18.3.2OHCI主机控制器驱动698
 - 18.4S3C2440/S3C6410 USB主机驱动的实现700
 - 18.5USB设备驱动701
 - 18.5.1USB设备驱动描述结构702
 - 18.5.2USB请求块URB703
 - 18.5.3URB的处理流程705
 - 18.5.4usb_bulk_msg()和usb_control_msg()708
 - 18.5.5探测和断开函数709
 - 18.6USB骨架程序710
 - 18.7USB设备驱动实例718
 - 18.7.1DNW驱动的实现718
 - 18.7.2USB键盘驱动的实现720
 - 18.8本章小结728
- 应用程序移植篇
- 第19章嵌入式Qt移植730
 - 19.1Qt开发环境搭建与使用 730
 - 19.1.1Qt SDK的下载与安装730
 - 19.1.2第一个Qt程序732
 - 19.1.3利用Qt Creator建立一个工程733
 - 19.2Qt的功能模块与裁剪735
 - 19.2.1Qt模块的构成735
 - 19.2.2图形用户界面736
 - 19.2.3信号与槽 736
 - 19.2.4布局管理737
 - 19.2.5主视窗737
 - 19.3嵌入式Qt移植与测试737
 - 19.3.1触屏库tslib移植737
 - 19.3.2Qt库移植739
 - 19.3.3嵌入式Qt程序测试741
 - 19.3.4嵌入式Qt工程配置与测试743
 - 19.4本章小结751
- 第20章嵌入式多媒体程序移植752
 - 20.1音频播放程序madplay的移植752
 - 20.1.1在Ubuntu中安装madplay752
 - 20.1.2 在Ubuntu中测试madplay753
 - 20.1.3将madplay移植到开发板753
 - 20.1.4在开发板中测试madplay755
 - 20.2视频播放程序MPlayer的移植755
 - 20.2.1在Ubuntu中安装MPlayer755
 - 20.2.2在Ubuntu中测试MPlayer756
 - 20.2.3将MPlayer移植到开发板756
 - 20.2.4在开发板中测试MPlayer758
 - 20.3利用Qt开发视频播放器758
 - 20.3.1MPlayer的SLAVE模式759
 - 20.3.2示例播放器MyPlayer的实现759
 - 20.4本章小结760
- 第21章嵌入式数据库移植761
 - 21.1SQLite数据库的使用761

21.1.1使用命令行操作SQLite761
21.1.2使用C语言操作SQLite763
21.1.3在Qt中操作SQLite765
21.1.4在QTableView中显示数据库内容767
21.2SQLite数据库的移植770
21.2.1将SQLite移植到开发板770
21.2.2在开发板中测试SQLite771
21.3本章小结773
第22章嵌入式Web服务器移植774
22.1Boa的使用与HTML页面测试774
22.2CGI程序测试779
22.2.1CGI的概念和原理779
22.2.2编写CGI脚本测试779
22.2.3CGIC库的基本使用780
22.3通过网页控制设备782
22.4通过网页监控设备785
22.5网页视频监控788
22.6将Web服务器移植到开发板791
22.6.1将Boa移植到开发板791
22.6.2将CGIC移植到开发板792
22.6.3将mjpg-streamer移植到开发板792
22.6.4在开发板上搭建Web服务站点793
22.7本章小结793
第23章嵌入式JVM移植794
23.1phoneME虚拟机移植794
23.1.1获取源代码795
23.1.2编译和安装795
23.1.3测试步骤796
23.2JamVM虚拟机移植798
23.2.1GNU Classpath移植798
23.2.2JamVM移植799
23.2.3JamVM测试799
23.3在JamVM上运行Jetty服务器800
23.3.1Jetty服务器启动800
23.3.2在Jetty服务器中部署Web应用801
23.3.3Servlet和JSP页面测试803
23.4本章小结805
• • • • • ([收起](#))

[深度实践嵌入式Linux系统移植_下载链接1](#)

标签

嵌入式

Linux

linux

评论

移植bsp的可以看看，但是已经过时。

还不错

有些代码走不通。另外，这本书只是描述了特定版本移植的具体过程，并没有总结一些思路让我们可以移植其他版本

[深度实践嵌入式Linux系统移植_下载链接1](#)

书评

[深度实践嵌入式Linux系统移植_下载链接1](#)