维普资讯 http://www.cqvip.com

基于不同类型Flash-ROM的Bootloader设计

张起贵,裴 科,张 刚,赵哲峰
ZHANG Qi-gui,PEI Ke,ZHANG Gang,ZHAO Zhe-feng

太原理工大学 信息工程学院,太原 030024
College of Information Engineering,Taiyuan University of Technology,Taiyuan 030024,China
E-mail:imaupk@163.com

ZHANG Qi-gui,PEI Ke,ZHANG Gang,et al.Design of bootloader based-on different flash-ROM.Computer Engineering and Applications,2007,43(33):112-114.

Abstract: Embedded microprocessor inner resources are more and more complicated so that the resourceful embedded systems development platform does not work efficiently without the management of operating system.How to develop BootLoader on the basis of the specific hardware platform is a key point and difficulty without question.The main similarities and differences of NAND Flash are compared with that of NOR Flash.The concept,function and implement analysis of embedded operating system booting program—Bootloader are introduced at different firmware-class storage device.Taking loading ARM-Linux operation system kernel as an example,software designing methods with two kinds of Flash-ROM——NAND and NOR Flash are expatiated in TI TMS320DM6446-based Embedded Development Platform

Key words: Bootloader;TMS320DM6446;NOR Flash;NAND Flash;embedded system

摘 要:嵌入式微处理器片内资源越来越复杂以至于离开操作系统管理的嵌入式系统将无法有效运行。如何根据开发板硬件资源来设计嵌入式操作系统的引导程序(Bootloader),无疑是一个重点和难点。介绍了具有不同固态存储类设备的嵌入式开发平台上的Bootloader的概念、主要任务和实现分析等。对比了NAND Flash和NOR Flash的主要异同。以加载ARM-Linux操作系统内核为例,阐述了基于TI TMS320DM6446嵌入式系统开发平台上利用NAND及NOR Flash两类Flash-ROM(闪存)引导加载操作系统的软件设计方法。

关键词: Bootloader;TMS320DM6446;NOR Flash;NAND Flash;嵌入式系统
文章编号:1002-8331(2007)33-0112-03 **文献标识码:**A **中图分类号:**TP368.1

1 引言

在嵌入式系统的开发中,嵌入式软件是实现各种系统功能的关键,也是计算机技术最活跃的研究方向之一。在专用的嵌入式开发板上运行操作系统(如Linux)以满足应用软件需求已经变得越来越流行,而Bootloader就是为引导操作系统内核运行的一段代码。通过它可以初始化硬件设备、建立或检测内存空间的映射图,其功能有点类似于PC机的BIOS(基本输入输出系统)程序[1]。它的主要作用是为运行操作系统提供基本的运行环境,并将操作系统的内核装载到存储器(RAM)中的合适位置上去运行。本文将以美国德州仪器(TI,TEXAS INSTRUMENTS)公司的DVEVM(DaVinci Evaluation Module)评估板为硬件平台,以TI公司的ARM+DSP开发软件CCSv3.2(Code Composer Studio)为开发调试环境,以ARM-Linux交叉编译工具cross-arm_v5t_le为编译器,通过分析比较NOR和NAND两类闪存的硬件功能特性以及DVEVM的启动机制,阐述Bootloader的运行原理与实现方法。

2 系统组成

典型的ARM嵌入式系统硬件平台一般包括一个以ARM为内核的处理器、存储器和必要的外部接口与设备,而在一些高端的(如数字信号处理、多媒体处理、低功耗及高性能)嵌入式平台上则会集成一个或多个DSP(Digital Signal Processor)核及其它特定功能的协处理器来提高性能和实用价值。在系统中,采用了内嵌ARM926EJ-S和TMS320C6446 DSP的TI公司的DM6446(全称TMS320DM6446)双核处理器,ARM端处理器主频294 MHz,CPU内置8 KB ROM,16 KB SRAM,8 KB指令Cache及16 KB的数据Cache,外部存储器使用256 MB的DDR2型SDRAM,64 MB的NAND Flash及16 MB的NOR Flash,外设包括DDR2控制器、ATA硬盘控制器、RS-232标准接口、USB2.0高速(480 Mbits/s)接口、10/100 M自适应以太网接口、LCD接口、CCD和CMOS图像接口、VLYNQ™接口及IEEE 1149.1 JTAG标准扫描接口等[2];DSP端处理器主频594 MHz,其它性能指标不属本文讨论范围,感兴趣的读者可参考

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60372058,No.60772101)。

作者简介:张起贵(1963-),男,副教授,主要研究领域为语音编码,嵌入式系统;裴科(1980-),男,博士生,主要研究领域为实时操作系统,嵌入式系统及其应用;张刚(1953-),男,博士,教授,博士生导师,主要研究领域为语音编码,嵌入式系统;赵哲峰(1975-),男,博士生,助教,主要研究领域为语音编码,嵌入式系统。

维普资讯 http://www.cqvip.com

相关文档了解更多信息。

软件平台由以下部分组成:Bootloader、嵌入式操作系统内核(Kernel)、文件系统(File system)。其中,嵌入式操作系统内核是嵌入式系统加电运行后的管理平台,负责实时性任务和多任务的管理。文件系统是嵌入式系统软件平台占用存储量最大的一部分,也是与用户开发最相关的一部分。它存储了系统配置文件、系统程序、用户应用程序等[3]。

3 Bootloader 固件程序设计

在开发基于 DM6446 处理器的嵌入式系统时,以何种方式引导启动系统是一个首先要考虑的问题。庆幸的是,DM6446处理器 ARM 端提供了各种各样的启动方式,总体上可分为从异步的外部存储器接口 AEMIF(Asynchronous External Memory Interface)/NOR Flash 引导启动和从 ARM 内部 ROM(AIROM,ARM Inner ROM)引导启动两种情况。这两种引导启动模式取决于 Boot Selection(BTSEL[1:0])引脚的设置状态。

BTSEL[1:0]=01 时,系统加电/复位后 DM6446 的 ARM 处理器将从 AEMIF 或 NOR Flash 零地址(0x02000000)处执行引导启动代码(由用户生成、下载和烧录),即从外部存储器启动;

BTSEL[1:0]≠01 时,系统加电/复位后 ARM 处理器首先从 AIROM 地址(0x00004000)处开始执行由芯片厂商在生产芯片时嵌入的 RBL(ROM Boot Loader)程序。DM6446 AIROM 固化的这段 RBL 代码支持以下 3 种不同的引导启动模式[4]:

BTSEL[1:0]=00,NAND Flash 引导启动模式:不能支持完整的固件程序启动,而是发送控制信息到用户编写的 NAND Flash 用户 Boot loader—UBL(User Boot Loader)并从 NAND Flash 拷贝 UBL 到 ARM 的内部 RAM(AIRAM,ARM Inner RAM)中执行,关于 UBL 的作用及实现方法后文详细叙述;

BTSEL[1:0]=11,UART 引导启动模式:同样不支持完全的固件程序启动,而是要通过 UART 下载串口 UBL 并发送控制信息到用户软件。几乎所有的嵌入式系统的 Bootloader 程序都实现了串口引导启动功能,所以关于 UART 启动本文不再讨论;

BTSEL[1:0]=10,保留未用。

不论采用哪种启动方式,系统加电或复位后,所有的 CPU 通常都从某个由 CPU 制造商预先安排的地址上取指令,首先执行的就是 Bootloader 程序。DM6446 处理器根据 BTSEL[1:0]引脚设置状态的不同来选择执行不同的 Boot Loader 代码,将根据以上提到的各种情况分别讨论相应 BootLoader 的不同特点及其实现过程。图 1 为 DM6446 的引导启动流程示意图。

3.1 NOR 与 NAND 简述

DaVinci EVM 评估板支持从 NOR 和 NAND 两类 Flash-ROM 引导启动操作系统,由于 NOR 与 NAND 型 Flash 在硬件特性等方面存在较大差别,所以用户所要编写的 Bootloader 也有一定的差异。

Flash-ROM 已经成为了目前最成功、最流行的一种固态内存,与 EPROM、EEPROM 相比具有读写速度快,而与 SRAM 相比具有非易失、以及价廉等优势。基于 NOR 和 NAND 结构的闪存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(后将该技术无偿转让给韩国 Samsung 公司),强调降低每比特的成本,更高的性能,并且像磁盘一样可以通过接口轻松升级。但是经过了十多年之后,仍然有许多业

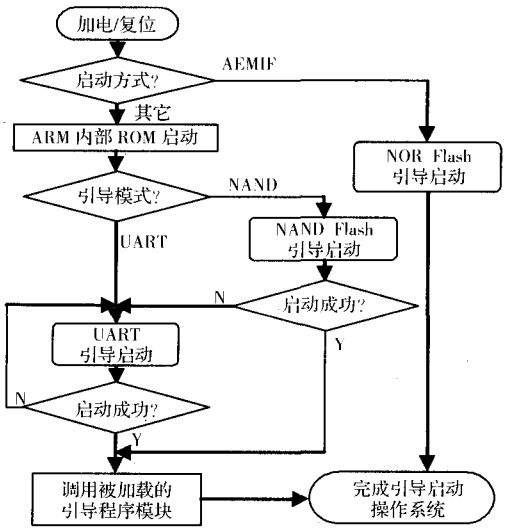

图 1 DM6446 引导启动流程图

内人士搞不清楚 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[5]。

NOR 的主要特点是芯片内执行 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 MB~4 MB 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。这也是为何所有的 U 盘都使用 NAND 闪存做为存储介质的原因。应用 NAND 的困难在于 Flash 的管理和需要特殊的系统接口。

篇幅所限,关于二者的性能比较、接口差别、可靠性等方面的详细介绍请参见文献[4]。

3.2 外部存储器启动模式

由于 Bootloader 的实现严重依赖于 CPU 的体系结构及板上硬件资源,因此 Bootloader 功能的实现基本可分为 Stage1 和Stage2 两大部分,分别运行于系统的 ROM 和 RAM 中[6]。当 DaVinci EVM 采用外部存储器启动模式时,由于 NOR Flash 具有 XIP 的特点,系统上电后首先从 NOR Flash 零地址开始执行引导启动程序 U-Boot 中的 Stage1 阶段代码,Stage1 除了完成中断向量的安装、特权模式下工作、禁止所有中断、堆栈指针寄存器初始化、外部 DDR2 型 SDRAM 存储器以及调试串口等的初始化外,主要负责把紧跟其后的 Stage2 部分从 NOR Flash 中拷贝到 DDR2 中指定地址并跳转到 Stage2 的 C 程序入口处继续运行,以提高系统的运行效率和实现更多的功能。最后由 Stage2 部分来完成操作系统的载入和启动。外部存储器启动模式下相对比较容易实现操作系统的引导加载,从功能上讲,与 NAND Flash 启动的第三阶段基本相同,详见下文叙述。有关 U-Boot 功能及移植方面的实例及文章已经非常多见,本文不再讨论。

3.3 AIROM 启动模式

DaVinci EVM 采取 AIROM 启动方式时,由于 DM6446 处理器 AIRAM 空间只有 16 KB 及 NAND Flash 不支持 XIP 技术等原因,要想完成从 NAND Flash 引导启动操作系统的任务则

维普资讯 http://www.cqvip.com

至少需要执行三个阶段的代码,它们所处位置如图2所示。其中虚线框内的部分是NAND引导启动的核心技术部分。Stage1:DM6446处理器AIROM代码,当开发板加电或执行复位操作后,处理器执行它内部的RBL代码;Stage2:用户Boot Loader代码(UBL),UBL被烧录在NAND Flash从Block1开始的5个存储块(Blocks)中,RBL将UBL从NAND拷贝到处理器AIRAM中并运行。所以UBL的大小一定要小于AIRAM的空间大小,说明文档推荐UBL不大于14KB,笔者的系统中所编译出来的UBL大小介于12.6 KB—14.2 KB之间均可以完成引导启动第三阶段代码的功能;Stage3:U-boot(与外部启动模式代码基本相同,包括两部分)是一个功能强大的Bootloader,除了实现与CPU体系结构相关的关键设备初始化功能外、还实现了读写DDR2、FLASH、串口下载、以太网口TFTP协议下载、传递内核参数以及命令行模式的用户交互界面等复杂的功能。它被存储在NAND Flash紧跟UBL代码之后,由UBL将其从NAND拷贝到DDR2种运行;Stage4:操作系统代码,例如ARM-Linux内核,存储于NAND Flash中U-boot之后,由U-boot将其从NAND拷贝到DDR2指定地址运行。至此,整个引导启动过程执行完毕,并将控制权交给操作系统。

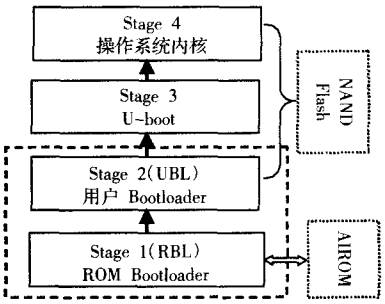

图2 NAND启动阶段代码框图

尽管DM6446处理器在生产时已经内嵌了第一阶段的代码RBL,但是它的内部实现机理对用户严格保密。RBL仅提供给用户一个编程接口,要求程序设计人员严格按照所提供的接口描述来编写出第二阶段的代码UBL,此段代码一般不对用户公开。因此,UBL在整个Bootloader的设计中处于十分重要的位置,起着非常重要的桥梁纽带作用,它一方面要和RBL接口交互,另一方面要为第三阶段的功能更加强大的Bootloader(U-boot)建立合适的运行环境。UBL实现的技术要点包括:

(1)设置引导配置寄存器BOOTCFG,使得BTSEL[1:0]=00,即选择了NAND Flash引导启动模式;并根据外部NAND Flash的型号来设置EM_WIDTH位,因为当RBL从NAND读取数据时会根据BOOTCFG寄存器的第5位(EM_WIDTH)的值来判断总线宽度和存取数据的大小,EM_WIDTH=0,数据宽度为8 Bit;EM_WIDTH=1时则为16 Bit;

(2)定义NAND UBL描述符(包括魔方数、UBL的入口点地址、页数、起始块号、起始页号等信息)。RBL首先要读取NAND Flash的设备ID,RBL代码中以设备信息表的方式定义了所有必需的设备存取信息(如Block,Page大小,UBL标识等),接着它会从NAND设备CIS/IDI Block之后(Block 1)的第0页开始读取有效的UBL描述符。若没有找到正确的能够与默认UBL标识相匹配的UBL,继续搜索下1个Block,最多可以连续搜索5个Blocks以避免NAND设备开始连续的几块有坏块的情况(事实上对所有的NAND设备来说几乎不会出现此类情况),如果仍然没有找到合适的UBL则返回到UART启动模式[3];

(3)当RBL找到一个有效的UBL时,首先要读取和处理带有加载、控制传输UBL等所必要信息的UBL描述符,RBL将根据UBL描述符里的信息来决定是否开启更快速的EMIF(External Memory Interface)或I-Cache(Instruction-Cache)等的组合操作,另外,UBL描述符也可设置在拷贝UBL时是否采用DMA方式。一旦具备了用户相关代码的运行条件,RBL将拷贝UBL到AIRAM中,并跳转到起始入口地址0x00000020处运行UBL;

(4)UBL主程序的主要任务是为第三阶段的功能完善的Bootloader建立运行环境。UBL主要功能函数按代码执行流程依次包括:初始化硬件平台DM644xInit();屏蔽所有中断,电源/休眠管理控制器复位,调用到的子函数有串口初始化设置UARTInit(),系统锁相环PLL1设置PLL1Init(),DDR锁相环PLL2设置PLL2Init(),DDR2初始化DDR2Init(),AEMIF接口初始化AEMIFInit(),中断向量表安装函数IVTInit()等;当读取到启动模式配置寄存器(BOOTCFG)值,如果BTSEL[1:0]=00则调用NAND初始化化函数NAND_Init()及拷贝代码函数NAND_Copy()将存储在NAND中紧跟其后的U-boot代码拷贝到DDR2中,如果拷贝成功则根据UBL中NAND启动参数结构体entryPoint成员的值来跳转到用户应用程序(U-boot)入口点运行。NAND启动参数结构体如下:

```
typedef struct {
Uint32 magicNum;    /* 期望的魔方数,有效值为0xA1ACEDxx 详见文献[3] */
Uint32 entryPoint;  /* 用户应用程序(U-boot)入口点 */
Uint32 numPage;     /* 存储U-boot所占页数 */
Uint32 block;   /* 存储U-boot的起始块号 */
Uint32 page;    /* 存储U-boot的起始页号 */
Uint32 ldAddress;   /* U-boot被加载到DDR2中的地址 - XIP模式 */
} NAND_BOOT;
```

至此,UBL的使命也就完成了,并将控制权交给了具有多重下载接口及引导操作系统等功能的U-boot。

为了更直观地了解RBL、UBL、U-boot及操作系统内核所处的位置及各自的作用,下面以图示方式给出DM6446从AIROM启动时,各个阶段之间的关系及执行流程,如图3所示。

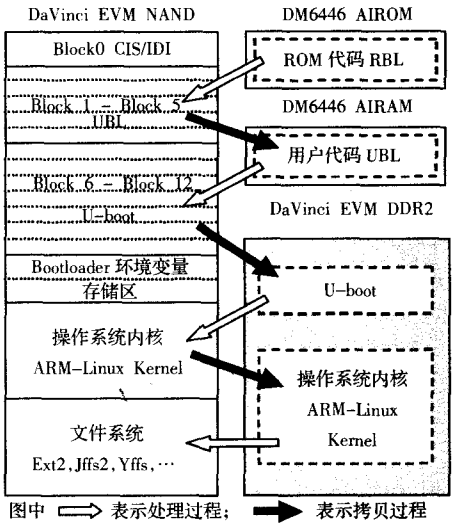

图3 AIROM NAND启动过程示意图

(下转166页)

维普资讯 http://www.cqvip.com

(5)比较过滤模块

本模块通过对上述各模块的处理结果进行操作,以达到过滤页面的效果。本文采用了多层次过滤的模式,首先采用黑名单地址过滤方法,对于非法网页进行排除;然后应用关键词过滤,进行初步的判断;最后将待分类文档的向量表示,转化到分类目标向量的空间上,然后按照3.2所述的方式,采用式(2)的改进Sim方法进行整个网页非法屏蔽或页面中部分不感兴趣内容的屏蔽过滤。

4.2 系统测试及结果分析

为了测试加入模糊评判后的过滤效果,从人民网、sina、163、21cn等几个网站对应的栏目下载了1 570篇新闻,其中包括政治方面268篇、经济方面201篇、军事方面582篇、体育方面249篇和计算机方面270篇,共5类作为实验数据。将这些数据取60%作为训练集,40%作为测试集。在一台2.53 GHz CPU,512 M内存,80 G硬盘,Windows 2000 professional操作系统的机器上,带宽为1 M的网络环境下进行实验。实验方法为:由训练集得到特征项(实验时关键词的选取采用过滤系统自动选取和人工整理相结合的方法),通过访问Web服务器的方式访问测试集进行测试。

针对α的不同取值对系统的过滤效率进行测试,其结果如图3所示。实验数据表明,当α取值0.79时,平均准确率(0.543)最高。

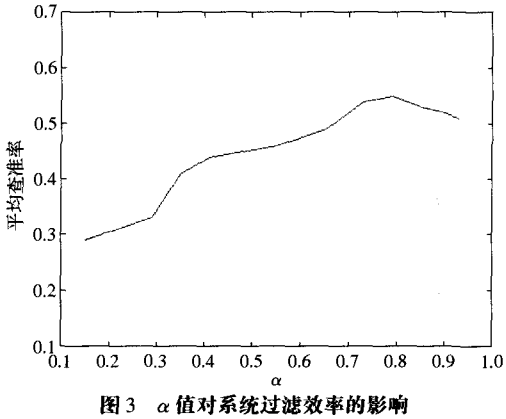

图3 α值对系统过滤效率的影响

通过对上述的网络信息过滤系统与该系统采用原Sim时的运行效果进行测试比较,经过多次测试,得到P-R图4。其中虚线为采用改进的Sim后的过滤效果,实线为采用原Sim时的过滤效果。从图中可以明显地看出,采用改进的Sim的系统的准确率比采用原Sim的系统有了一定的提高。实验数据显示在分类准确率方面平均高出9.62%。

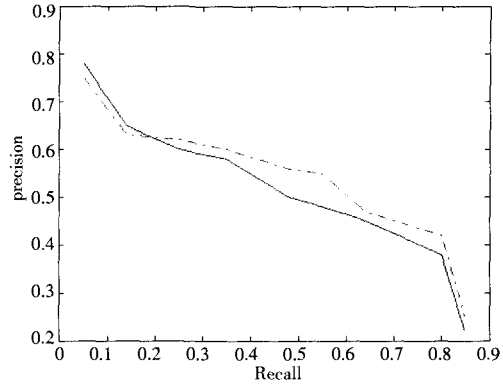

图4 两种系统效果比较P-R图

5 结束语

本文初步探讨了模糊评判和向量夹角余弦相结合的一种信息空间相似度Sim的计算方法。首先运用潜在语义索引的方法得到目标信息空间的矩阵表示,然后运用模糊评判和向量夹角余弦两种相结合而改进了的Sim,求待分类空间与其的相似程度。通过这种方法进行信息过滤,考虑了部分精确度不高的特征对过滤结果的影响,提高了网络信息过滤的准确率。但是也正因此,原Sim方法在查全率上要优于文中改进的Sim方法。另外,文中的改进方法仅仅是在已确定特征项及其权值的情况下,对相似度计算的改进,因此改进的效果受所选择的特征项及其权值的影响。(收稿日期:2007年5月)

参考文献:

[1] 张宁,贾自艳,史忠植.使用KNN算法的文本分类[J].计算机工程,2005,31(8):171-185.

[2] 袁方,杨柳,张红霞.基于k-近邻方法的渐进式中文文本分类技术[J].华南理工大学学报:自然科学版,2004,32(z1):88-91.

[3] 阮彤,冯东雷,李京.基于贝叶斯网络的信息过滤模型研究[J].计算机研究与发展,2002,39(12):1564-1571.

[4] Yang Qing,Li Fang-min.Support vector machine for customized e-mail filtering based on improving latent semantic indexing[C]//Proceedings of the Fourth International Conference on Machine Learning and Cybernetics,Guangzhou,18-21 August 2005.

[5] 何新贵.模糊知识处理的理论与技术[M].2版,北京:国防工业出版社,1997:425-428.

[6] 叶浩,王明文,曾雪强.基于潜在语义的多类文本分类模型研究[J].清华大学学报:自然科学版,2005,45(9):1818-1822.

(上接114页)

4 结束语

Bootloader是依赖于硬件而实现的,每个目标板的硬件配置都不完全一样,因此Bootloader程序也都不会完全一样。本文以TI的DaVinci EVM评估板为硬件平台,阐述了具有两类Flash-ROM,多种启动模式下的Bootloader运行的主要步骤和关键技术,为实现引导Linux操作系统内核运行提供合适的环境。两类Flash-ROM在嵌入式系统中的应用和实现,可满足不同使用场合和实验环境的用户需求。常见的资源简单的嵌入式系统都采取了从NOR型Flash启动的方式,本文重点讨论了如何构建一个基于NAND型Flash引导操作系统的Bootloader。此外,设计与实现一个功能强大的优秀的Bootloader程序是一个庞大复杂的过程,在程序中如能多利用CCS所提供的各类工具会是帮助我们调试的好方法。(收稿日期:2007年8月)

参考文献:

[1] Lennon A.Embedding Linux[J].IEEE Review,2001,47(3).

[2] Texas Instruments Incorporated.TMS320DM6446 Digital Media System-on-Chip.Reference Guide[Z].SPRS283D.Texas,USA,2006.

[3] 刘晶晶.基于ARM-Linux嵌入式系统引导程序的设计[J].微计算机信息,2006,22(2-2):123-125.

[4] Texas Instruments Incorporated.TMS320DM644x DMSoC ARM Subsystem.Reference Guide[Z].SPRUE14.Texas,USA,2005.

[5] Arie Tal.NAND和NOR flash技术设计师在使用闪存时需要慎重选择[J].今日电子,2002,4(1):12-13.

[6] 詹荣开.嵌入式BootLoader技术内幕[EB/OL].[2007-07-23].http://www-128.ibm.com/developerworks/cn/linux/l-btloader/index.html#author.