第 36 卷 第 2 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.2 Computer Engineering January 2010
·软件技术与数据库· 文章编号：1000—3428(2010)02—0033—03 文献标识码：A 中图分类号：TP311

无线传感器网络终端引导程序设计

刘 浩[1]，邹雪城[1]，李 鹏[2]，刘 明[1]，任红强[1]
(1. 华中科技大学电子科学与技术系，武汉 430074；2. 珠海高凌信息科技有限公司，珠海 519060)

摘 要：针对无线自组织对等网络的结构特点及对无线可移动终端的需求，分析 Linux 操作系统的启动过程，给出无线传感器网络可移动终端 5 段式引导程序的设计方法，对引导程序实现的 4 个关键环节的配置和设计进行说明。调试结果表明引导程序可成功地运行在自主设计的无线终端硬件平台上。

关键词：自组织网络；无线传感器网络；移动终端；引导程序

BootLoader Design for Terminal in Wireless Sensor Network

LIU Hao[1], ZOU Xue-cheng[1], LI Peng[2], LIU Ming[1], REN Hong-qiang[1]
(1. Department of Electronic, Science and Technology, Huazhong University of Science and Technology, Wuhan 430074;
2. Zhuhai Comleader Information Science and Technology Co., Ltd., Zhuhai 519060)

【Abstract】 Focusing on the structural characteristics of the wireless self-organization network and the requirements of wireless mobile terminal, a five-stage BootLoader design approach for mobile terminal based on Linux in wireless sensor network is derived. The approach can make starting flow of BootLoader program design more clearly. This paper analyzes configure and design methods of four key issues in BootLoader program implementation. The results show that the 5-stage BootLoader program can normally run in the self-design terminal platform.

【Key words】 self-organization network; wireless sensor network; mobile terminal; BootLoader

1 概述

对等网络(Peer-to-Peer, P2P)[1-2]和自组织网络(self-organization network)[3-4]是目前国际计算机网络技术领域的研究热点，有别于传统通信网络的 Client/Server 机制，对等网络节点之间不仅可以直接通信，而且每个节点都可作为中间节点为其他节点提供服务，使本不能相互覆盖的 2 个或多个网络节点之间实现通信与数据传输。无线传感器网络作为新一代的传感器网络，充分借鉴了对等网络技术和自组织网络技术的特点。因此，美国商业周刊和 MIT 技术评论在预测未来技术发展的报告中，分别将无线传感器网络列为 21 世纪最有影响力的 21 项技术和改变世界的 10 大技术之一。终端作为网络的实体和业务的承载体，节点芯片是整个无线传感器网络的基础，网络及其关键技术的研究应首先搭建网络和业务的承载平台，可移动终端则成为验证节点芯片移动性、数据传输、覆盖范围等性能的平台。基于该思路，本文重点阐述了无线传感器网络移动终端引导程序(BootLoader)的设计实现。

适用于终端的嵌入式操作系统主要包括 Symbian, Windows Mobile, PALM OS48 和 Linux。由于 Linux 具有源代码的开放性和内核的可配置性等特点，因此本设计选择内核版本 2.4 的 Linux 作为终端的操作系统。自主设计的移动终端硬件平台主要由 ARM9 嵌入式处理器、射频单元(RF)、存储体、音频处理、触摸式液晶屏控制、键盘输入和电源管理等单元构成，并内置以太网和 USB 接口。其中，存储体部分包含 CPU 片内 Flash、片内 SRAM、外置大页面 Nand Flash 以及高速低功耗 PSRAM(Pseudo SRAM)。

2 5 段式引导程序设计流程

BootLoader 是终端上电或复位之后先于操作系统内核运行的引导程序。引导程序 5 段式设计流程包括前期准备、初始化与参数配置、装载映像文件、内核的引导及系统初始化、Linux 内核启动。程序设计采用汇编语言与 C 语言混合方式：汇编部分实现 CPU 的初始化、存储空间初始化等；C 语言部分则完成加载模式的判决、内核映像文件装载等，其工作流程如图 1 所示。引导程序支持 2 种工作模式：加载模式和下载模式，其中，启动加载为默认模式。

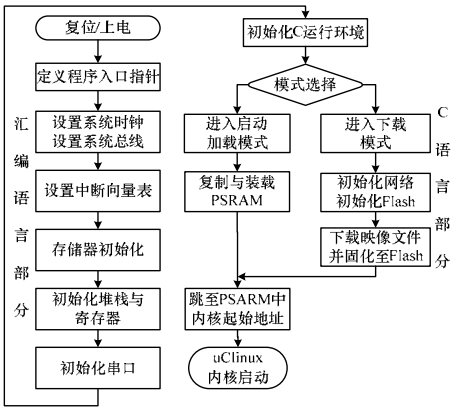


图 1 引导程序工作流程

2.1 准备阶段

前期准备阶段包括终端硬件平台设计、节点芯片驱动程序、大页面 Flash 的驱动程序设计、系统启动方式选择、Linux

基金项目：国家“863”计划基金资助项目(2006AA01Z226)
作者简介：刘 浩(1970－)，男，博士，主研方向：无线自组织网络，超大规模集成电路，片上网络；邹雪城，教授、博士生导师；李 鹏，工程师、硕士；刘 明，讲师、硕士；任红强，工程师、硕士
收稿日期：2009-06-20 **E-mail：**freemansoc@126.com

内核和文件系统映像文件的编译、内核加载方式配置、存储空间配置等工作。编译完成的引导程序和映像文件烧写至外部 Nand Flash。重新上电后，根据配置管脚的状态，处理器自动将引导程序的启动代码从 Nand Flash 前 4 KB 空间拷贝到处理器 Nand Flash 控制器内置 SRAM(Steppingstone)中运行，引导完成系统的初始化和映像文件的加载。

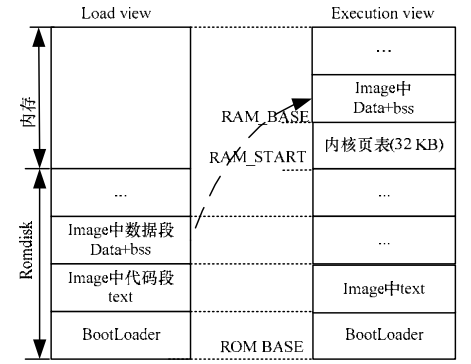
2.2 硬件系统初始化与参数配置阶段

该阶段工作是完成系统硬件部分的初始化，包括屏蔽所有的中断、设置 CPU 速度和时钟频率、存储体初始化、Nand Flash 初始化、GPIO 端口和 UART 初始化、关闭 CPU 内部指令/数据 Cache(如 CPU 不具备内部的数据/指令 Cache，其相关的函数返回值为 0)、定义程序的入口地址等。

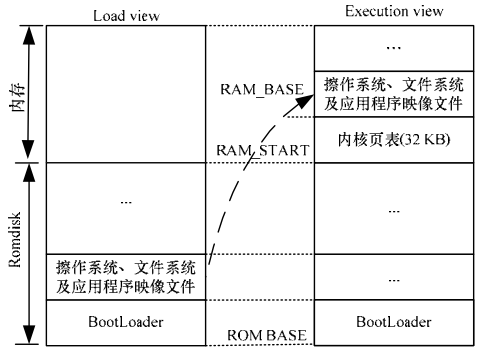
2.3 装载映像文件

在 PSRAM 中分配 128 KB 的单元作为 Ramdisk 系统，作为可读写数据段，建立一个内核的运行环境。然后将 Flash 中的映像文件装载到内存中，该内存单元作为 Romdisk 系统直接运行内核。同时需要将该单元保护起来，避免误操作或其他非法指令和地址修改内核部分的代码。

操作系统、文件系统和应用程序构成的映像文件有 2 种装载模式：Flash-resident Image 和 Flash-based Image。前一种是引导程序，仅仅把 Image 文件中的数据段(data＋bss)复制到系统内存中，代码段(text)在 Romdisk 中直接运行；后一种则是引导程序把 Image 完全复制到系统内存中执行，包括 Image 中的代码段(text)和数据段(data+bss)。图 2 描述 2 种模式下 Image 的内存地址空间分配及装载过程。本文采用图 2(b)所示的装载模式。



(a)Flash-resident 模式



(b)Flash-based 模式

图 2　Image 内存分配及其装载过程示意图

2.4 内核的引导及系统初始化

上述步骤完成之后，程序计数器指针(PC)跳转到内核起始地址处，完成内核解压、安装及其环境参数配置。设置体系结构环境，进行命令参数的解析，设置中断和异常向量表，进行进程调度器、定时器、控制台的配置、Cache 初始化、内存页面初始化、设备初始化等。操作系统的初始化还包含文件系统的安装，如 Ext2 文件系统、管理 Nand Flash 的 JFFS2 文件系统。

2.5 Linux 内核的启动

引导程序引导完成后释放对硬件系统的控制权，转交给 Linux 操作系统，并释放清除使用过的临时内存，然后跳转到操作系统内核(kernel)的第 1 条指令地址，启动 Linux 操作系统，执行/etc 目录下的用户系统配置信息，准备系统应用程序的使用环境。

3 引导程序设计实现

引导程序的实现包括 4 个关键环节的配置：内存规划，堆栈分配，中断向量配置及 Nand Flash 读写操作。

3.1 内存规划

内存规划包括 2 个方面：(1)内核映像所占用的内存范围；(2)根文件系统所占用的内存范围以及应用程序和程序申请的缓冲区所占用的内存。对于内核文件，将其拷贝到从 RAM_BASE(MEM_START+0x8000)这个基地址开始的大约 1 MB 的内存范围内，如图 2(b)所示。以 MEM_START 为基址的前 32 KB 内存需要空出，让 Linux 内核放置一些全局数据结构：启动参数和内核页表等信息。根文件系统映像文件则将其拷贝到以 MEM_START+0x100000 为基址的内存中(采用 Ramdisk 作为根文件的系统映像，其解压后的大小一般为 1 MB)。

3.2 堆栈分配

ARM 有 7 种工作执行状态，每一种状态的堆栈配置都是独立的，所以，对程序中需要用到的每一种模式都要为堆栈指针 SP(Stack Pointer)定义一个堆栈地址，使其指向该运行模式的栈空间。这样，当程序的运行进入异常模式时，可以将需要保护的寄存器放入 SP 所指向的堆栈，而当程序从异常模式返回时，则从对应的堆栈中恢复，采用这种方式可以保证异常发生后程序的正常执行。改变程序状态寄存器(CPSR)内的状态位(低 5 位)可使处理器切换到不同的工作状态，根据系统使用中断和异常的情况，可能需要初始化部分或全部堆栈指针寄存器。本文的堆栈配置包括外部中断模式、快速中断模式、系统调试模式、未定义指令模式、系统模式和用户模式。其中，外部中断模式栈配置程序如下：

```
InitStack
MOV   R0, LR
MSR   CPSR_c, #0xd2     //设置外部中断模式堆栈
LDR   SP, StackIrq
MOV   PC, R0
…
StackSvc DCD SvcStackSpace + (SVC_STACK_LEGTH - 1)* 4
StackIrq DCD IrqStackSpace + (IRQ_STACK_LEGTH - 1)* 4
…
SvcStackSpace SPACE SVC_STACK_LEGTH * 4
//管理模式堆栈空间
IrqStackSpace SPACE IRQ_STACK_LEGTH * 4
//中断模式堆栈空间
…
```

对管理模式堆栈而言，SP 的值由 SvcStackSpace 的地址加上 SVC_STACK_LEGTH 的大小而定。系统所有的堆栈均位于系统运行空间 PSRAM 中。可通过外部输入命令的方式切换工作模式，并通过查看特殊寄存器的内容帮助诊断系统

运行状态。

3.3 中断向量配置

异常中断向量表(Exception Vector Table)是 BootLoader 与 Linux 内核发生联系的关键所在，当 Linux 内核得到处理器的控制权，一旦发生中断，处理器将自动跳转到中断向量表中的某个表项(根据异常、中断类型)处读取指令运行。中断向量配置包含未定义指令、软中断、取指令中止、取数据中止、中断以及快速中断等。对于其中的未定义指令，中断配置如下：

```
interrupt vectors：
Reset
LDR    PC, ResetAddr          //加电、重启
LDR    PC, UndefinedAddr      //未定义指令
LDR    PC, SWI_Addr           //软中断
LDR    PC, PrefetchAddr       //取指令中止
LDR    PC, DataAbortAddr      //取数据中止
DCD    0xb9205f80             //保留
LDR    PC, [PC, #-0xFF0]      //IRQ 中断
LDR    PC, FIQ_Addr           //快速中断
ResetAddr    DCD ResetInit    //分配 ResetInit 地址
IRQ_Addr    DCD    0
FIQ_Addr    DCD    FIQ_Handler
```

本文所用 CPU 的中断系统提供了 32 个中断源，共分为 3 类：快速中断 FIR，向量化中断 Vector Interrupt 和非向量化中断 NoVector Interrupt。当硬件产生中断异常后，CPU 将跳转到 0x18 空间执行相应的异常处理函数去处理中断，在这个地址上，存放 1 条赋值语句：LDR PC, [PC, #-0xFF0]，执行 PC 赋值指令。这条指令将 VICVectAddr 的地址赋值给 PC，如果此时是 UART0 中断，则 VICVectAddr 中将是 VICVectAddr1 寄存器的内容。CPU 根据这个寄存器的内容执行相应的中断处理函数。

从“堆栈分配”内容可知，系统在 Vector_IRQ 中保存中断的堆栈及返回位置，并且进行模式的切换；然后跳入 IRQ_USR 或 IRQ_SVC 程序中执行，执行后返回被中断的程序继续执行。

3.4 大页面 Nand Flash 操作

移动终端除了满足系统自身对数据存储的需要，同时也为使用者以及分布式数据存储提供大容量存储空间，所以，系统中采用大容量大页面的 Nand Flash。本文选用 Flash 的页面容量为 4 KB+128 Byte，其中，主存储区为 4 KB，扩展区(Spare area)为 128 Byte。

由于目前很多 SJF(Sec Jtag Flash)工具均无法正确识别，因此，需要按照 Nand Flash 的结构特点配置读写模式。每个页面大小为 2 KB/4 KB，通过修改原 SJF 中的 Flash.c 文件，按照目标 Flash 修改厂商号与设备号，修改相应的操作命令，调整编程、擦除、读操作的地址周期，就可以实现对大页面 Flash 的正确操作。

在设计可读写文件系统时将 4 KB+128 Byte 大页面等幅分割成 8 个 512 Byte +16 Byte 的小页面进行处理，并禁止操作系统对 Flash 进行错误检测(基于 Flash 器件出厂无坏块)。待文件系统解压成功后，坏块检测功能交由文件系统管理。采用等幅分割的方法不仅解决了当前文件系统对大页面 Nand Flash 支持不好的问题，在支持大页面 Flash 的情况下，也兼容小页面的 Flash。

当引导程序最后一条指令完成后，跳转到内核的第 1 条指令地址 MEM_START+0x8000 启动内核，引导程序的任务全部结束，Linux 操作系统开始管理终端系统，至此，已经为后续驱动程序调试以及业务开发作好了准备。

4 结束语

从调试打印信息中可以看出，Linux 版本号、CPU 识别信息、时钟配置、内存空间配置以及外设初始化信息等显示全部正确，表明了 5 段式引导程序能够成功地运行于自主设计的无线移动终端硬件平台上，完成了映像文件的加载、解压，操作系统能够开始正常运行。

参考文献

[1] 唐九阳, 张维明, 肖卫东, 等. 类人类社会基于社区的对等网自组织构造[J]. 计算机研究与发展, 2006, 43(8): 1383-1390.

[2] Fox G. Peer-to-Peer Networks[J]. Web Computing, 2001, 3(3): 75-77.

[3] Pottie G J, Clare L P. Wireless Integrated Network Sensors: Towards Low Cost and Robust Self-organizing Security Networks[C]//Proc. of SPIE Conf. on Sensors, C3I, Information, and Training Technologies for Law Enforcement. Orlando, FL, USA: [s. n.], 1999: 86-95.

[4] 林晓帆, 李 超. 基于 P2P 网格高效广播传递算法的研究[J]. 计算机工程, 2007, 33(7): 101-103.

编辑 张正兴

(上接第 27 页)

5 结束语

本文提出了基于 BF 路由表的非结构化 P2P 网络搜索改进算法，通过在节点上构建路由表，使得节点能够了解一定范围内的节点资源共享信息，减少了查询消息盲目发送的可能性，在一定程度上实现了针对性查询。仿真实验表明，该算法极大地减少了 Gnutella 网络中查询消息的通信量，并能够获得较高的查全率。由于网络中节点互换基于 BF 的路由表是本算法实现的基础，路由表的互换会产生一定的网络通信流量，因此采用优化机制尽可能减少网络通信流量以及处理 P2P 网络节点动态性对路由表的影响是本文下一步的研究目标。

参考文献

[1] Matei R, Ian F, Adriana I. Mapping the Gnutella Network: Properties of Large-scale Peer-to-Peer Systems and Implications for System Design[J]. IEEE Internet Computing Journal, 2002, 6(1): 50-57.

[2] Christos G, Milena M, Amin S. Random Walks in Peer-to-Peer Networks[C]//Proc. of the IEEE INFOCOM'04. New York, USA: IEEE Press, 2004: 120-130.

[3] Crespo A, Garcia-Molnia H. Routing Indices for Peer-to-Peer Systems[C]//Proc. of the 22nd IEEE International Conference on Distributed Computing Systems. Vienna, Austria: [s. n.], 2002.

[4] Joseph S R H. Neuro Grid: Semantically Routing Queries in Peer-to-Peer Netwokrs[C]//Proc. of International Workshop on Peer-to-Peer Computing. Pisa, Italy: [s. n.], 2002.

[5] Bloom B. Space/time Trade-offs in Hash Coding with Allowable Errors[J]. Communications of the ACM, 1970, 13(7): 422-426.

编辑 索书志