基于 Linux 的无线传感器网络引导程序的设计

针对无线传感器网络的结构特点及对无线可移动终端的需求，分析了 Linux 操作系统的启动过程，提出了无线传感器网络可移动终端引导程序的设计方法，并对引导程序实现的 4 个关键环节的配置和设计进行了说明。实际调试结果表明：

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。

0 引言

对等网络(Peer-to-Peer,P2P) 和自组织网络(SelforganizationNetwork) 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。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。在实际应用中，基于 ARM 处理器和嵌入式技术的无线传感器网络系统在环境监测、医疗监护等领域得到了广泛的应用。

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。所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 NandFLASH 以及高速低功耗 PSRAM（Pseudo SRAM）。

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。BootLoader 与硬件息息相关，硬件环境不同，BootLoader 也不同，要建立一个通用的 BootLoader 几乎是不可能的。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。

1 引导程序设计流程

引导程序设计流程包括系统配置、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：其中，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，图 1 所示是其工作流程图。引导程序支持加载模式和下载模式两种工作模式，其中，启动加载为默认模式。

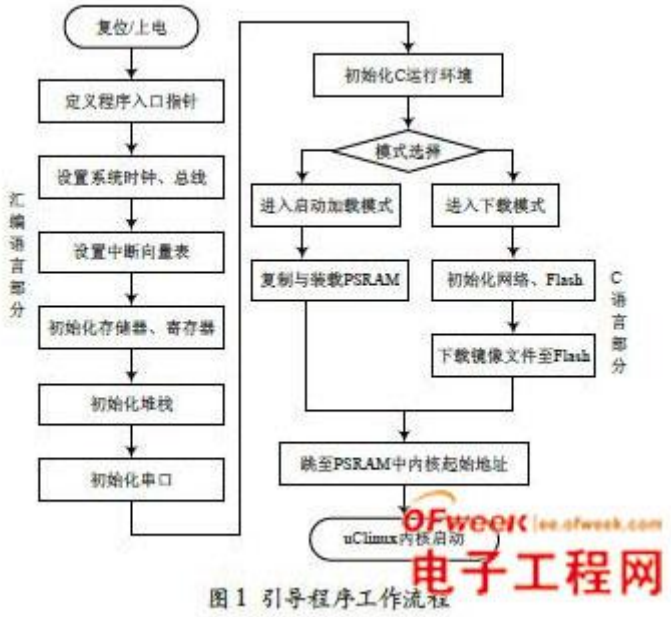

图 1 引导程序工作流程

1.1 系统配置

系统配置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 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可烧写至外部 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，同时引导完成系统的初始化和镜像文件的加载。

1.2 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、NandFlash 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。

1.3 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两种装载模式：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）。

1.4 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。

1.5 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。

2 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。

2.1 内存规划

内存规划包括两个方面：第一是内核映像所占用的内存范围；第二是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+0×8000） 这个基地址开始的大约 1 MB 的内存范围内，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，如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+0×100000 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）。

2.2 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

…

St a ckSvc DCD SvcSt a ckSpa c e+（SVC_ STACK_

LEGTH-1）*4

St a c k I r q DCD I r qSt a c k Sp a c e + （ 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 结语

本文提出无线传感器网络可移动终端引导程序的设计方法，从实际调试看，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，表明了采用该方法设计的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。