文章编号:2096-1618(2018)04-0395-05

基于嵌入式 Linux 内核的自恢复设计

张 杰[1], 杨笔锋[1,2], 马尚昌[1,2], 严学阳[1]
(1.成都信息工程大学电子工程学院,四川 成都 610225;2.中国气象局大气探测重点开放实验室,四川 成都 610225)

摘要:随着嵌入式 Linux 设备应用越来越广泛,在使用过程中却经常发现由于 NAND FLASH 制造工艺的缺陷造成系统内核丢失或者是内核损坏并且设备出现突然不能够启动的现象。讨论基于嵌入式 Linux 内核的自恢复设计,将系统内核进行备份,在系统启动失败的时候通过 U-boot 自动恢复系统内核。Linux 内核备份是在 NAND FLASH 上面设置一个内核备份区并烧写一个内核。设计原理是修改 U-boot 代码在系统启动的时候验证系统内核并判断内核的正确性,如果系统错误就调用备份区内核并覆盖错误的系统从而恢复系统内核。设计解决了嵌入式 Linux 系统丢内核的难题。

关 键 词:Linux 内核;U-boot;自动恢复;内核备份
中图分类号:TP368.1 **文献标志码**:A
doi:10.16836/j.cnki.jcuit.2018.04.008

0 引言

现在很多设备将 Linux 内核放在 NAND FLASH 上面,由于制造的缺陷 NAND FLASH 会经常发生位翻转或者坏块现象或者对 NAND FLASH 的不当操作等造成内核丢失系统不能正常启动的现象[1-2]。现在各个行业对设备的稳定性提出了越来越高的要求,保障设备的安全性和稳定性一直是嵌入式 Linux 行业努力的方向。Linux 内核的自恢复设计主要是为了解决因 NAND FLASH 发生位翻转或者坏块造成系统内核遭到破坏引起的设备不能正常启动的问题。采用迈冲科技 ARM 开发板开发环境是 CPU:AT91SAM9G20、U-boot-1.3.4 、linux-2.6.30。U-Boot 提供了板级初始化、网络通信、引导系统等功能,通过修改 U-boot 和 Linux 内核相关代码利用在 NAND FLASH 备份区的内核恢复遭到破坏的系统内核,大大减少了设备出现故障的概率,通过系统恢复使设备的安全性和稳定性得到提高[3-4]。想要系统能够进行自动恢复前提条件是修改 U-boot 启动第二阶段的代码使之能够调用备份内核覆盖原来的内核;修改内核代码设计一个内核备份区存储内核备份;内核不能启动的时候 U-boot 能够运行。以下情况需要进行系统自动恢复:内核不能完全启动;内核启动后出现异常;NAND FLASH 存储内核的区域出现多个 Bit 位翻转;NAND FLASH 出现内核区域小规模坏块现象。设计解决了一定条件下系统不能正常启动的问题,对提高设备运行稳定性有一定的价值。

收稿日期:2017-11-23
基金项目:四川省教育厅重点资助项目(16ZA0210)

1 U-boot 引导启动流程介绍

U-boot(universal boot loader)是嵌入式操作系统广泛应用的开源的引导程序,经过不断地发展已经变得功能强大支持多种操作系统和处理器。设计使用的 U-boot 版本是 U-boot-1.3.4,对嵌入式 Linux 系统的引导分为两个阶段 stage1 和 stage2。U-boot 引导程序就像电脑的 BIOS 程序一样对嵌入式 Linux 系统进行引导,是设备启动时最先运行的代码。所以 U-boot 在 FLASH 存储的地方是第一个分区。U-boot 的源代码和 Linux 内核非常相似其中有很多代码都是从 Linux 内核移植过来的尤其是驱动[5-8]。分析 U-boot 的引导过程就得找出程序的入口以及各个代码的功能,以下是 U-boot 引导过程的两个阶段[9-10]。

1.1 U-boot 引导程序第一阶段

Stage1 是一些非常精简的汇编代码包括在 u-boot-1.3.4\board\atmel\at91rm9200dk\u-boot.lds、u-boot-1.3.4\cpu\arm920t\start.S 和 u-boot-1.3.4\cpu\arm920t\at91rm9200\lowlevel_init.S 几个文件中。在 u-boot.lds 文件中指定了以 ENTRY(_start)为起始入口,而它指向了 start.S 文件。start.S 指向 start_armboot 也就是第二阶段代码的入口。start.S 调用 lowlevel_init.S 用于芯片存储器的初始化执行完后返回[11-12]。

start. S 是 Stage1 的主要部分,其主要功能如下:(1)定义程序入口;(2)对 CPU 进行一些初始化;(3)将 stage2 的代码拷贝到 RAM 空间;(4)设置好堆栈;(5)引导跳转到第二阶段的入口。

1.2 U-boot 引导程序第二阶段

Stage2 都是 C 语言代码编写,其主函数是 lib_arm\board. c 里面的 start_armboot 函数也是程序的入口[13]。u-boot-1. 3. 4\common\main. c 里面的 main_loop 是 U-boot 循环命令调用函数。Stage2 包含了板载的基本驱动如:LED、NAND FLASH、1-wire、spi 等。驱动程序和 Linux 内核的驱动非常相似,很多驱动都可以从内核移植过来。由 main_loop 调用环境变量里面的命令或者是用户输入的命令最后交给 run_command 函数来执行,包括 NAND FLASH 相关的命令。上电后程序执行流程如图 1 所示[14]。

Stage2 是 U-boot 的关键,主要功能如下:(1)初始化相关的硬件设备(如:NAND FLASH、LED);(2)CPU 时钟初始化;(3)完成内存映射;(4)将内核读到内存中,从 NAND FLASH 中读取内核到 RAM 中;(5)设置启动参数(如:环境变量);(6)调用内核并传递启动参数。Stage2 的 C 代码功能强大、结构复杂、难度大,设计需要理解底层代码的原理与代码的架构。

2 总体方案设计

2.1 方案介绍

嵌入式 linux 内核启动流程如图 1 所示。

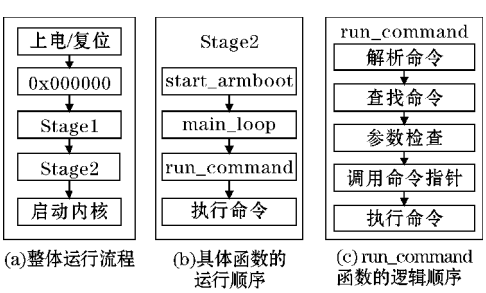

图 1 嵌入式 linux 内核启动流程

run_command 函数是执行命令的入口函数。U-boot 的命令执行实际上就是 main_loop 函数将环境变量里面的命令传递给 run_command 函数然后 run_command 函数解析参数然后通过指针调用各种命令的函数指针。设计将更改 run_command 函数,它调用了 U-boot 的各种命令包括:NAND FLASH 命令、环境变量存储的命令、引导启动命令等[15]。

U-boot 会对内核进行校验,校验涉及的各种函数都有返回值,利用返回值从而引导进行内核自动恢复。嵌入式 linux 内核自恢复设计方案如图 2 所示。

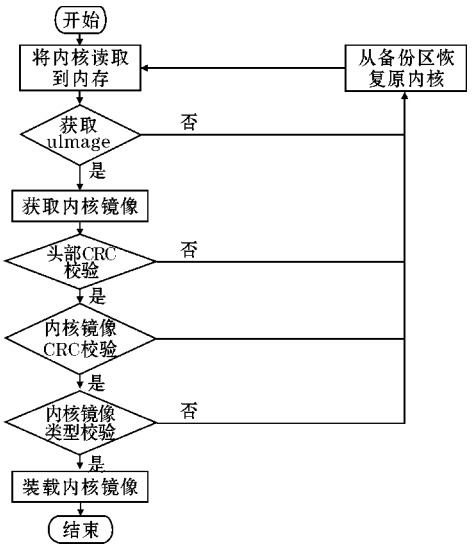

图 2 方案设计

方案设计实际上就是将读取到的内核进行各种校验然后判断内核的完整性,如果内核完整就启动内核,如果内核不完整就通过备份内核覆盖原来的内核,具体介绍如下:

(1)将内核读取到内存实际上就是用 nand read 命令将 NAND FLASH 指定地址上的 uImage 读取到内存的指定地址上。nand 的各种命令详见 common\cmd_nand. c 里面的 U_BOOT_CMD 表。

(2)获取 uImage 是执行 bootm 命令具体是执行 do_bootm 函数,实际上 do_bootm 函数是启动整个系统的入口它获得 uImage 镜像头部信息,并对 uImage 镜像的类型进行检查。

(3)获取内核镜像以及内核镜像头部指针是 boot_get_kernel 函数。boot_get_kernel 函数实际上是查找一个完整的 zImage 镜像。

(4)image_get_kernel 函数将对内核镜像头部进行校验和对整个内核镜像进行校验,并返回内核镜像的头部指针。

(5)如果通过以上函数判断出内核不完整就通过调取备份区的内核将原内核覆盖然后重新进入启动流程。

2.2 内核备份区设计

采用 128 M 的 NAND FLASH 内核分区设计和 NAND FLASH 烧写地址如图 3 所示,通过 cat /proc/mtd 命令得到的结果如图 4 所示。设计使用的 nandflash_at91sam9g20ek. bin、u-boot. bin、uImage、rootfs. jffs2 文件

大小分别是4 KB、158 KB、1733 KB和5760 KB。开发者可用通过计算所烧写的文件大小来自己确定各分区大小或者文件烧写地址。

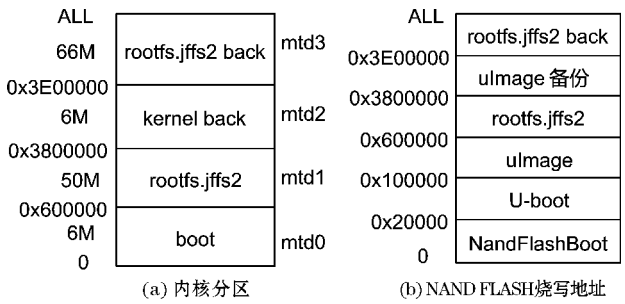

图 3　内核分区和 NAND FLASH 烧写地址

图 3(a)是 Linux 内核的 4 个分区,其中 boot 分区包含 NandFlashBoot、U-boot、uImage 3 部分,rootfs. jffs2 分区是文件系统,kernel back 分区是备份的 uImage,rootfs. jffs2 back 分区是备份的文件系统。图 3(b)是各个文件在 NAND FLASH 上面的具体烧写地址。内核启动后通过 cat /proc/mtd 命令可以查看内核对 NAND FLASH 的分区信息。由图 4 可以看出各个分区的大小以 16 进制显示,每个分区的最小擦除大小是 0x20000 单位 Bit 换算成 10 进制就是128 KBit。

```
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00600000 00020000 "boot"
mtd1: 03200000 00020000 "rootfs.jffs2"
mtd2: 00600000 00020000 "kernel back"
mtd3: 04200000 00020000 "rootfs.jffs2 back"
#
```

图 4　内核分区截图

2.3　代码修改

2.3.1　内核分区代码修改

内核分区代码是在/arch/arm/mach-at91/目录下 board-sam9g20ek. c 文件里面的 ek_nand_partition[ ]结构体数组。结构体数组包含分区名、地址偏移指针、分区大小 3 个元素,分区名用字符串表示地址和分区大小用十六进制表示。各个参数可以参考图 3(a)。

2.3.2　U-boot 代码修改

如图 1 所示,U-boot 进入 main_loop 主循环后有几秒钟等待用户按键,如果用户没有按键将进入自动启动阶段。自启动阶段的参数由环境变量提供。

要修改的实际上是 run_command 函数。run_command 函数的第一个字符串参数传递命令,函数返回 1 表示命令执行完成可重复,返回 0 表示命令执行不可重复,返回 -1 表示命令未执行或者命令参数错误。*cmdtp指针变量将指向具体命令函数的指针。将 if((cmdtp->cmd)(cmdtp,flag,argc,argv)! =0){rc=-1;}这句代码修改为:

```
    if(cmdtp->cmd ! =do_bootm)
  {
  if((cmdtp->cmd)(cmdtp,flag,argc,argv) ! =0){
    rc=-1;}
  }else if(cmdtp->cmd == do_bootm)
    {
    if((cmdtp->cmd)(cmdtp,flag,argc,argv) ==1){
  rc = -1;
  printf("faild to bootm so get the new kernel form kernel back area. \n");
  run_command("nand erase 0x100000 0x200000",0);
  run_command("nand read 0x22200000 0x3800000 0x200000",0);
  run_command("nand write 0x22200000 0x100000 0x200000",0);
  run_command("boot",0);
  }
  }
```

修改代码的目的就是发现内核出错,然后通过备份内核覆盖它并重新启动系统。

3　设计验证

3.1　假设只烧写备份区内核

当只烧写备份区内核时设备串口输出如图 5 所示。如果没有内核的自动恢复设计,在内核分区没有系统内核设备将不能够启动,或者是在内核分区 NAND FLASH 在有数据的地方出现小部分坏块设备也不能够启动。有了内核自动恢复设计 U-boot 将调用备份区的内核覆盖掉原来的内核,如果有小部分坏块就会跳过该坏块,使得系统恢复正常。

```
Reading data from 0x2ff800 -- 100% complete.
 2097152 bytes read: OK
Wrong Image Format for bootm command (1)
ERROR: can't get kernel image!
faild to bootm so get the new kernel form ker(3) back area.

NAND erase: device 0 offset 0x100000, size 0x200000
Erasing at 0x2e0000 -- 100% complete.
OK

NAND read: device 0 offset 0x3800000, size 0x200000
 2097152 bytes read: OK
```

图 5　只烧写备份区内核时设备启动串口输出

图 5 中标记①显示内核镜像格式错误,因为此时读取 FLASH 区域里面没有内核。所以接下来输出找不到内核镜像(标记②)从内核备份区获取新的内核(标记③)。最后将输出“Loading Kernel Image…OK.”表示内核完整装载。

3.2 假设内核出现损坏

使用软件 UltraEdit 将 uImage 修改一个字节或者一块区域,如图 6 所示。将地址130 h的头两个字节 7F 修改为 00。通过这种方法模拟出的环境和实际 NAND FLASH 出现位翻转的情况是一样的,用相关工具读取出来的数据也是一样的。如果没有内核的自动恢复设计,在内核分区 NAND FLASH 出现多个 Bit 位翻转设备将不能够启动,或者是一块数据出错设备也不能启动。

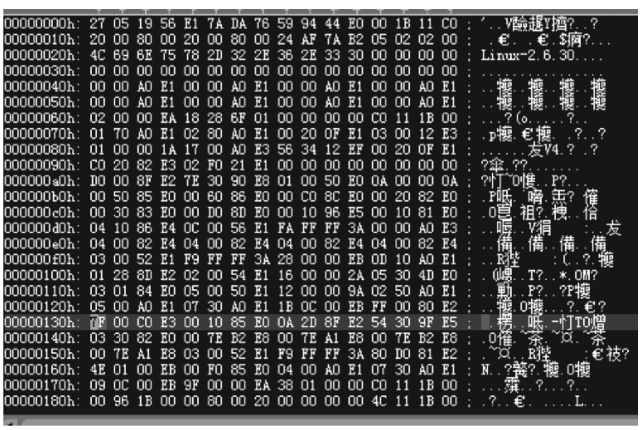

图 6 修改 uImage

```
Reading data from 0x2ff800 -- 100% complete.
 2097152 bytes read: OK
## Booting kernel from Legacy Image at 26200000 ...
   Image Name:   Linux-2.6.30
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1774016 Bytes =  1.7 MB
   Load Address: 20008000
   Entry Point:  20008000
   Verifying Checksum ... Bad Data CRC (1)
ERROR: can't get kernel image!
faild to bootm so get the new kernel form kernel back area.

NAND erase: device 0 offset 0x100000, size 0x200000
Erasing at 0x2e0000 -- 100% complete.
OK

NAND read: device 0 offset 0x3800000, size 0x200000
 2097152 bytes read: OK

NAND write: device 0 offset 0x100000, size 0x200000
 2097152 bytes written: OK

NAND read: device 0 offset 0x100000, size 0x200000

Reading data from 0x2ff800 -- 100% complete.
 2097152 bytes read: OK
## Booting kernel from Legacy Image at 26200000 ...
   Image Name:   Linux-2.6.30
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1774016 Bytes =  1.7 MB
   Load Address: 20008000
   Entry Point:  20008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK (2)
OK
```

图 7 损坏内核时设备启动串口输出

将修改的 uImage 烧写在原内核区将完整的 uImage 烧写在内核备份区,并重启设备观察串口输出如图 7 所示。

图 7 标记①显示 U-boot 通过 CRC 校验成功检测出了内核错误,并通过接下来的命令用备份区域的内核将错误的内核覆盖掉最终从新启动设备如标记②显示,由此可以看出恢复系统成功。

4 结束语

以软件修改二进制内核模拟 NAND FLASH 损坏的环境进行实验,通过实验验证得出的结果来看本方案设计是成功的,能够一定程度自动恢复损坏的系统内核。目前各种设备的稳定性越来越受到重视,本设计能大幅提高设备的稳定性,不需要硬件的修改只需修改相关代码,有很好的实用价值。

参考文献:

[1] 郭明,侯彦华. 嵌入式 Linux 系统的现状与未来[J]. 北京广播电视大学学报,2003(3):37-40.

[2] 汤瑞. 一种基于 NAND Flash 固态硬盘的坏块管理方法[J]. 电子科技,2014,27(8):40-42.

[3] 周军. NAND Flash 的坏块管理设计[J]. 单片机与嵌入式系统应用,2010(9):15-17.

[4] 邢晓敏. 嵌入式系统的发展与应用[J]. 中国水运,2011,11(6):68-70.

[5] 蒋光明. 基于嵌入式 Linux 的无线视频监测系统的研制[D]. 合肥:安徽大学,2015.

[6] 刘志超,荆琦. Linux 内核函数调用关系的验证方法[J]. 微型机与应用,2014(21):4-5.

[7] 段红祥. 面向 MES 终端的嵌入式 Linux 平台研究和设计[D]. 重庆:重庆大学,2008.

[8] 吴明晖. 基于 ARM 的嵌入式系统开发与应用[M]. 北京:人民邮电出版社,2004.

[9] 李飞. 一个嵌入系统的设计[J]. 成都信息工程学院学报,2003,18(3):232-235.

[10] 史永宏,耿增涛,张守伟. 基于状态恢复的嵌入式 Linux 快速引导[J]. 计算机工程与设计,2008,29(2):277-278.

[11] 金刚,吴军,马鹏,等. 嵌入式 Linux 系统移植中 SMP 的实现研究[J]. 信息技术,2016(8):93-96.

[12] 李革梅. 嵌入式 linux 操作系统裁剪和定制研究[D]. 太原:中北大学,2005.

[13] 陈军,涂亚庆. 加快嵌入式 Linux 系统启动速度

的方法及应用[J]. 后勤工程学院学报,2005,21(3):54-58.
[14] 霍妍,孟凡荣. 基于 Linux 嵌入式系统的研究与实现[J]. 计算机系统应用,2004,13(8):4-6.
[15] 孟雷,忽海娜,MENGLei,等. ARM-Linux 嵌入式系统 BootLoader 的配置与移植[J]. 计算机技术与发展,2008,18(10):204-206.

Self-recovery Design of Embedded Linux Kernel

ZHANG Jie[1], YANG Bi-feng[1,2], MA Shang-chang[1,2], YAN Xue-yang[1]
(1. Chengdu University of Information Technology;2. The Key Laboratory of China Meteorological Administration,Chengdu 610225,China)

Abstract: As embedded Linux device has been widely used,however,it is often found that the system kernel core is lost or damaged and the equipment suddenly can not start because the defects of the NAND FLASH manufacturing. This article discusses a self-healing design based on the embedded Linux kernel, and the functions of it are to backup the system kernel and automatically restore the system kernel via U-boot when the system fails to boot. Linux kernel backup is to set up a kernel backup area in NAND FLASH and write a kernel. The design principle is to modify the U-boot code to verify the system kernel and judge the correctness of the kernel when the system is starting. If the system is in error, the kernel of the backup partition is called to cover the wrong system to restore the kernel of the system. This design solves the problem of dropping kernels in embedded Linux systems.

Keywords: Linux kernel; U-boot; self-recovery; kernel backup