DOI：10.16652/j.issn.1004-373x.2010.22.038

ARM-Linux 嵌入式系统的 Boot Loader 分析与设计

王世阳，余学才，梁锡宁，陈　涛，朱良销，苏　柯
（电子科技大学 光电信息学院，四川 成都　610054）

摘　要：Boot Loader 作为 ARM 嵌入式系统的一个重要部分。对于使用不相同版本的内核的系统板，所对应的 Boot Loader 也是相同，因此对每个系统板都要对其运行其所对应的 Boot Loader。在此简要介绍 S3C2410 及其两种启动方式，着重分析 S3C2410 从 NANDFLASH 启动的过程中，对各个内部功能模块进行初始化过程，并设计出基于 S3C2410 嵌入式系统的 Boot Loader。通过在线仿真及实际测试表明，该 Boot Loader 具有良好的稳定性、实时性和可移植性。

关键词：ARM；Boot Loader；嵌入式系统；启动方式
中图分类号：TN911-34；TP311　　**文献标识码**：A　　**文章编号**：1004-373X(2010)22-0071-03

Analysis and Design of Boot Loader Based on ARM-Linux Embedded System

WANG Shi-yang，YU Xue-cai，LIANG Xi-ning，CHEN Tao，ZHU Liang-xiao，SU Ke
（SOEI，University of Electronic Science & Technology of China，Chengdu 610054，China）

Abstract：Boot Loader is an important part of ARM embedded system. For different kernel system board，its Boot Loader is also different. Each bare-board should program its own Boot Loader. Therefore，the development of specific Boot Loader is particularly important，because the superiority of Boo Lloader directly affects on the performance of embedded systems. The S3C2410 and its two start-up modes are introduced. The process of initialization that S3C2410 makes for each internal module is analyzed emphatically during the start-up of NANDFLASH. A Boot Loader based on S3C2410 embedded system is designed. The online simulation and practical tests show that the Boot Loader has good stability，real-time performance and portability.

Keywords：ARM；Boot Loader；embedded system；start-up mode

0　引　言

由 Boot Loader 和固化在固件(firmware)中的 Boot 代码(可选)共同组成一个嵌入式系统的引导加载程序。它的作用和功能就像固化到计算机内主板上的一个 ROM 芯片程序 BIOS(basic input output system)。但是它一般不配置像 BIOS 那样的固件程序，这是因为要考虑经济方面的原因，因此必须自己完成这方面的工作。Boot Loader 可以初始化硬件设备，建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境。它的实现严重地依赖于硬件，特别是在嵌入式系统中，即使基于同一个 CPU 的 Boot Loader，对于不同的板子，也有很大的不同[1]。

1　Boot Loader 分析

系统加电，然后复位后，基本上所有的 CPU 都是从复位地址上取得指令的。以微处理器为核心的嵌入式系统中，通常都有某种类型的固态存储设备(FLASH，E^2PROM 等)，这个固态存储设备被映射到一个预先设置好的地址上。在系统加电复位后，一开始处理器就会去执行存放在复位地址处的程序，而且通过开发环境可以将 Boot Loader 定位在复位地址一开始的存储空间上，因此 Boot Loader 是系统加电后，在操作系统内核或者一些应用程序被运行之前，首先会运行的程序。对于嵌入式系统来说，比较复杂的或者为了方便后期开发大的应用程序，有的使用操作系统，也有很多的情况下，因功能简单，或仅包括应用程序的系统不使用操作系统，但是不论有无操作系统在启动时都必须执行 Boot Loader，为的是准备好软硬件运行环境。

以微处理器为核心的嵌入式系统中，一般都有某种类型的固态存储设备(FLASH，E^2PROM 等)，这个固态存储设备被映射到一个预先设置好的地址上。在系统加电复位后，一开始处理器就会去执行存放在复位地址处的程序。而且通过开发环境可以将 Boot Loader 定位在复位地址一开始的存储空间上，因此 Boot Loader 是系统加电后，在操作系统内核或者一些应用

收稿日期：2010-06-16
基金项目：国家“863”计划资助项目(2010AAJ206)

程序被运行之前,首先会运行的程序。对于 Linux 系统,它的主要任务有以下 7 个方面。

(1) 初始化处理器及外设的硬件资源配置。一般嵌入式系统的处理器在上电复位后,外部的 I/O 引脚都处于输入状态,处理器的片内和片外设备资源都需要配置。

(2) 建立内存空间的映射图,从而将系统的软硬件环境带到一个合适的环境,这样就能为最终启动操作系统的内核提供最好条件。

(3) 把操作装载到映射的内存中,这也是所有任务当中最重要的一个,只有完成这个任务,操作系统才能被装载到内存当中去,Boot Loader 一般会提供串口和网络装载两种方式。

(4) 为了把操作系统的映像保存在 FLASH 中,以便以后启动,可以直接装载 FLASH 的数据,而不用重新下载程序,但需要对 FLASH 进行编程。

(5) 运行操作系统。设置相关的寄存器和资源,跳转到操作系统的所在空间,进行相关的引导,这就是 Boot Loader。

(6) 在 Linux 系统启动时,传递系统的启动参数,可以给内核传递命令行等参数,通过命令行可以选择控制系统的启动模式。

(7) 命令行的解析和输入/输出控制。为了开发的方便,多数的 Boot Loader 都采用串口作为终端的控制方式[2]。

Boot Loader 的启动过程可分为两个重要阶段。第一阶段:由于 Boot Loader 的实现依赖于 CPU 的体系结构,所以设备代码的初始化等功能都在该阶段完成。而且,为了达到缩短代码的目的,通常用汇编语言来编写。在这一阶段的执行过程中,又可分为几个方面。

① 硬件设备的初始化。在该阶段的执行过程中,首先需要对硬件设备进行初始化,其目的主要是为第二阶段的执行以及随后 Kernel 的调用准备基本的硬件环境。

② 为加载 Boot Loader 的第二阶段准备 RAM 空间。为了获得更快的执行速度,通常把第二阶段加载到 RAM 空间中来执行。因此,必须为加载 Boot Loader 准备好一段可用的 RAM 空间范围。

③ 设置堆栈指针。设置堆栈是为了执行 C 语言代码作好准备。

④ 跳转到第二阶段的 C 入口点。当程序执行到这个位置时,可以通过修改 PC 寄存器的值,使其跳转到第二阶段。

第二阶段阶段的启动流程分析:为了便于实现复杂的功能和获得更好的代码可读性和可移植性,通常第二阶段的代码用 C 语言来实现。但是,与普通 C 语言的不同之处是,这里使用了“弹簧床”的概念,即先用汇编语言写一段小程序,并将这段小程序作为第二阶段可执行映像的执行入口点,然后在汇编程序中用 CPU 跳转指令跳入 main ( )函数中去执行,当 main ( )函数返回时,CPU 执行路径再次返回到汇编程序中第二阶段,包括初始化本阶段要使用的硬件设备,检测系统内存映射,会将 Kernel 映像和根文件系统映像从 FLASH 中读到 RAM 空间中,为内核设置启动参数调用内核。

2 Boot Loader 的设计

2.1 中断向量表(二级)的设计与建立

如果有中断或者异常发生时,处理器便会强制性地把 PC 指针指向向量表中它所对应的中断类型地址值。为了提高中断响应速度,FLASH 的 0_X0 地址存放能跳转到 $0_X33FFFF00$ 地址处中断向量的跳转指令,也就是会在在 RAM 中建立一个二级中断向量表,起始地址为 $0_X33FFFF00$。除了复位外,所有的异常入口地址都由 FLASH 跳转得到,代码如下:

```
#define _ISR_STARTADDRESS (SDRAM_END- 0x100)
                                        //0x33FFFF00
definepISR_RESET (* (unsigned *)(_ISR_STARTAD-
DRESS+0x0))                             // x33FFFF00
#define pISR_UNDEF (* (unsigned *)(_ISR_STAR-
TAD-DRESS+0x4))                         // x33FFFF04
```

2.2 第二阶段拷贝到 RAM

把第二阶段 Stage2 拷贝到 RAM 地址的最顶大小为 1 MB 的开始空间, RAM 的起始地址为 $0_X30000000$。代码如下所示[3]:

```
/* 计算在FLASH中的位置,假设该映像不超过 64K,可
修改该值 */
Adr r0,_start
Add r2,r0,#(64 * 1024)
Add r0,ro,#0x1000
Ldr r1,BLOB_START
/* 开始复制 stage2 到 RAM,R0=源起始地址,R1=目的
地址,R2 源结束地址 */
copy_loop:
ldmia r0!,{r3- r10}
stmia r1!,{r3- r10}
cmp r0,r2
ble copy_loop
ldr r0,BLOB_START
```

2.3 堆栈指针的设置

用户使用哪些中断决定了系统堆栈的初始化,以及系统需要处理的哪些错误类型。一般情况下,堆栈设置是必须,而且是由管理者自己设置的。如果需要使用 IRQ 中断,那么 IRQ 堆栈的设置也是必须的,下面是 IRQ 堆栈的设置[4]:

```
IRQMode //堆栈
orr r1,r0,#IRQMODE|NOINT
msr cpsr_cxsf,r1; IRQMode
ldr sp,IRQStack
```

3 Stage2 的设计

3.1 可执行映像 Stage2 的入口

由于 Glibc 库支持的函数不能用于编译和链接 Boot Loader 这样用 C 语言编写的程序,因此把 main()函数的起始地址作为第二阶段的入口点是最直接的想法。可以用汇编编写一段 Trampoline 小程序,用 CPU 跳转指令跳到 main()函数去执行,当函数返回时会再次回到 Trampoline 程序[5],代码如下:

```
ldr sp DW_STACK_START @setup stack pointer
mov fp,#0 @no previous frame,so fp=0
mov a2, #0 @set argv to NULL
bl main @call main
mov pc,#FLASH_BASE @otherwise,reboot
```

程序顺利时就不会再回到开始的 Trampoline 程序,不然就会回到最后的语句,系统就会重新启动。

3.2 内存影射

一般 S3C2410 上配置的 SDRSAM 大小为 64 MB,该 SDRAM 的物理地址范围是 0x30000000 ~ 0x33FFFFFF(属于 Bank 6)。由 Section 的大小可知,该物理空间可被分成 64 个物理段。因为 ARM 体系结构中数据缓冲必须通过 MMU 开启,因此 Boot Loader 效率不是很高,但是 MMU 可以通过平板映射(虚拟地址和物理地址相同)方式被开启,这样使用内存空间 Dcache,从而使 Boot Loader 的运行速度得到有效的提高[6]。映射关系代码如下[7-8]:

```
void mem_mapping_linear(void)
{ unsigned long descriptor_ index, section_ base, sdram_ base,
sdram_ size;
sdram_ base=0x30000000;
sdram_ size=0x4000000;
for (section_ base =sdram_ base,descriptor_ index =
sec-tion_ base>>20;Ssection_ base<sdram_ base+
sdram_ size; rdescrip-tor_ index +=1; section_ base +=
0x100000)
{* (mmu_ tlb_ base +(descriptor_ index)) =(section_ base
>>20) |
MMU_ OTHER_ SECDESC;}
}
```

3.3 装载内核映像和根文件系统映像

像 ARM 这样的嵌入式 CPU 通常都是在统一的内存地址空间中寻址 FLASH 等固态存储设备的,因此从 Flash 上读取数据与从 RAM 单元中读取数据一样,用一个简单的循环就可以完成从 FLASH 设备上拷贝映像的工作:其中 count 为根文件系统映像的大小或内核映像的大小[9]。

```
While(count)
{ *dest++=*src++;//src 为 FLASH 中的地址, dest 为 RAM 中的地址
count--=4;
}
```

3.4 内核的启动参数的设置

内核启动可以从 NAND FLASH(NOR FLASH)中启动运行 Linux,需要修改启动命令如下[10]:

```
#ifdef CONFIG_S3C2410_ NAND_ BOOTChar Linux_
cmd [ ] = " noinit root =/dev/bon/2 init =/Linuxrc console=
tty0 console=ttys0" ;
#else
CharLinux_ cmd[]="CharLinux_ cmd[]=" noinit root=
/dev/bon/3init =/Linuxrc console=tty0 console=ttys0" ;
```

LCD 启动参数一般都包括 root,init 和 console。noinitrd 不使用 ramdisk。root 根文件系统在 MTD 分区。Init 内核运行入口命令文件。consol 内核信息控制台,ttys0 表示串行口 0[11];tty0 表示虚拟终端。

4 结 语

通过对 Boot Loader 的分析可以看出,设计一个性能优良的 Boot Loader 可以提高系统的稳定性及实时性,它是嵌入式开发中不可或缺的一部分。只有设计出一个稳定的 Boot Loader,才能进行下一步的系统开发工作,直至完成整个嵌入式系统的开发。设计 Boot Loader 是一项很复杂的工作,需要对硬件资源和所用的操作系统有很深的理解。在实际开发中可以根据需要简化设计,去除不必要的系统功能,这样可以大大提高程序执行的效率和稳定性。这里给出的 Boot Loader 已经顺利通过了调试,可以正常加载操作系统。

参 考 文 献

[1] 马忠梅.ARM 嵌入式处理器结构与应用基础 [M].北京:北京航空航天大学出版社,2003.

[2] 李驹光.ARM 应用系统开发详解:基于 S3C4510B 的系统设计[M].北京:清华大学出版社,2003.

[3] 曹程远.U-Boot 在 S3C2410 上的移植[J].微型电脑应用,2005,21(7):48-50.

[4] 郭志,江秀臣,曾奕.一个嵌入式系统的启动分析[J].微计算机信息,2005,21(32):28-30.

[5] 陈海军,申卫昌,史颖.嵌入式系统引导程序详探[J].计算机技术与发展,2006,16(1):123-125,128.

[6] Microsoft Corporation. Microsoft extensible firmware initiative FAT32 file system specification [M]. 1st ed. [S.l.]: [s.n.], 2000.

(下转第 77 页)

间作为精确的发送时间戳 t_2，而主时钟接收到该报文时也记下接收时刻的精确时间戳 t_3，并将该事件戳在随后的延迟响应报文。中发送给从时钟节点。如图 6 所示。

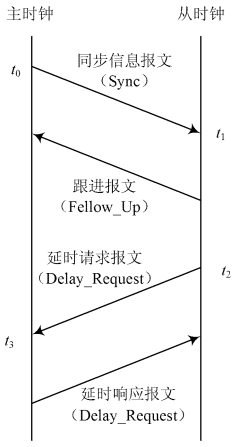

图 6 IEEE1588 报文发送示意图

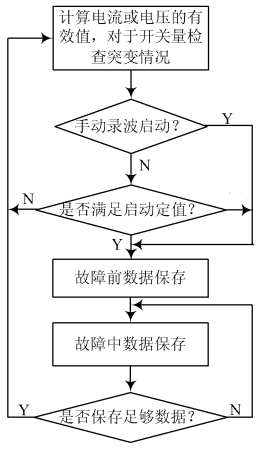

图 7 启动判断与存储的流程图

主、从时钟偏差(offset)以及网络延迟(delay)可表示为：

$$A = t_1 - t_0 = \text{delay} + \text{offset},$$

$$B = t_3 - t_2 = \text{delay} - \text{offset};$$

$$\text{delay} = (A + B)/2, \quad \text{offset} = (A - B)/2$$

4.6 故障录波启动判断及记录模块

因协议转换器已对数据加入时间戳并进行合并，故障录波启动判断及记录模块存在实时性的问题，设计时注重更大的系统容量，因此硬件平台选择 Intel CPU，软件基于 Linux 操作系统。它通过额外的算法判断同步的模拟量采样数据与开关量数据的瞬时值或有效值来判断当前电网中是否发生故障，需要高速存储并生成故障报告[10]。同时可在正常状态下存储常态录波。

5 结 语

新型故障录波器采用两层设计，对传统站与数字站进行了统一的封装，使得单一型号的录波器产品可以满足传统站，数字站以及传统数字混合站的要求，解决了当前过渡时期的多种要求，大大降低了录波设备的开发、生产和维护成本。同时，它同时支持大容量，高采样率的暂态故障录波需求和常态录波。在 96 路模拟量，192 路开关量的容量下，对于传统站可以支持达到 10 kHz的采样率，对于数字站可以支持4.8 kHz的采样率。它是一种高性能，实用性良好的新型故障录波器。

参 考 文 献

[1] 刘振亚.特高压电网[M].北京：中国经济出版社，2005.

[2] 中华人民共和国国家发展和改革委员会.DL/T 860.9-1(IEC 61850-9-1)变电站通信网络和系统 第 9-1 部分：特定通信服务映射(SCSM)-通过单向多路点对点串行通信链路的采样值[S].北京：中国电力出版社，2006.

[3] 中华人民共和国国家发展和改革委员会.DL/T 860.7-1(IEC 61850-7-1)变电站内通信网络和系统 第 1 部分[S].北京：中国电力出版社，2006.

[4] 顾永红，杨巧丽.基于 MPC8260 和 VxWorks 实现快速以太网通信[J].电子工程师，2008(1)：66-67.

[5] 王柯，邹锐，陈娟.数字化变电站故障录波器通信规约研究[J].贵州电力技术，2008(7)：26-28.

[6] WindRiver. Network protocol toolkit user's guide 5.5[M/OL]. [2004-12-25]. http://forums.windriver.com.

[7] 吴敏良，石旭刚，张胜，等.基于 IEEE1588 的同步以太网实现方式[J].单片机与嵌入式系统应用，2010(1)：38-40.

[8] 梁晓兵，周捷，杨永标，等.基于 IEC61850 的新型合并单元的研制[J].电力系统自动化，2007(31)：85-89.

[9] 宋丽军，王若醒，狄军峰，等.GOOSE 机制分析、实现及其在数字化变电站中的应用[J].电力系统保护与控制，2009(14)：31-35.

[10] 金华荣，苏茂钧，高建坤，等.一种新型数字化故障录波及分析装置的实现[J].电力系统保护与控制，2009(18)：116-119.

作者简介：王大千 男，1985 年出生，陕西安康人，在读硕士研究生。主要研究方向为嵌入式系统、网络通信。
周 余 男，1980 年出生，重庆人，讲师。主要研究方向为嵌入式系统、Linux 系统。
都思丹 女，1963 年出生，南京人，教授、博士生导师。主要研究方向为图像处理与模式识别、嵌入式系统，近场技术。

(上接第 73 页)

[7] SanDisk Corporation. Sandisk secure digital card product manual [M]. 2nd ed. [S.l.]: [s.n.], 2004.

[8] Sumsung Electronics. S3C2410X 32 b risc microprocessor user's manual[M]. [S.l.]: [s.n.], 2003.

[9] 霍拉鲍夫.嵌入式 Linux：硬件、软件与接口[M].陈雷，译.北京：电子工业出版社，2003.

[10] 徐睿，黄健，徐辰.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.

[11] 李善平.Linux 与嵌入式系统[M].北京：清华大学出版社，2002.

作者简介：王世阳 男，1983 年出生，山东人，硕士研究生。研究方向为数字图像处理、嵌入式软件开发。
余学才 男，1960 年出生，四川人，教授，硕士生导师。主要从事图像处理、模式识别、光学无损检测。
梁锡宁 男，1985 年出生，江苏人，硕士研究生。研究方向为数字图像处理、嵌入式软件开发。