嵌入式系统的内核引导启动过程浅析

仲崇权　杨素英　陈宝君
大连理工大学电子与信息工程学院

摘　要　引导加载程序(Bootloader) 是嵌入式系统 CPU 加电后第一个开始运行的代码。在内核映像执行之前完成相关的底层硬件的初始化，建立内存空间的映射图等重要工作，然后为内核提供引导参数，启动内核。通过对 Bootloader 的体系结构和工作机理进行深入研究，并结合德国 DENX 开发的具有功能强大的 Bootloader 的 U-BOOT 启动程序，给出 U-BOOT 在基于 AT91RM9200 处理器的嵌入式系统板上的启动过程。

关键词　嵌入式系统，Bootloader，U-BOOT

Abstract　**Boot Load Program (Bootloader) is the first running program after powering up an Embedded Board. Bootloader performs the specific hardware initialization and sets up the memory space map and other tasks before the Embedded Operating System kernel image executes. After that , it provides the boot parameters for the kernel and starts up the kernel. U-Boot is a powerful Bootloader. This paper analyses the U-Boot section the codes of startup process in detail in an Embedded System board based on the AT91RM9200 CPU.**

Keywords　**embedded system, Bootloader, U-BOOT**

1．引言

嵌入式系统和 Linux 的有机结合，成为后 PC 时代计算机最广泛的应用形式。一个嵌入式 Linux 系统从软件的角度看由四个部分组成：引导程序，Linux 内核，文件系统和用户应用程序。Bootloader 是引导程序其作用是初始化硬件设备，为最终调用操作系统内核做好准备。U-BOOT 就是一种 Bootloader，给操作系统提供一个标准的接口，屏蔽了硬件的多样性，因此减少了开发周期，同时支持 ARM 体系结构和 MIPS 结构[1]。

2．固态存储设备的典型空间分配结构

嵌入式系统的固态存储设备的典型空间分配结构可以映像 Bootloader、内核启动参数、内核映像和根文件系统为四个部分,如图 1 所示[2]。其中，BootLoader 就是在操作系统内核运行之前运行的一段小程序。通过这段小程序，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引导到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境。在嵌入式系统 Bootloader 是严重地依赖于硬件而实现的，每种不同的 CPU 体系结构以及不同的嵌入式板级设备的配置都有不同的 Bootloader。

图 1. 固态存储设备的典型空间分配结构

3．Bootloader 操作模式

大多数 Bootloader 都包含启动加载模式和下载模式两种不同的操作模式。

启动加载模式：Bootloader 在这种模式下从目标机上的某个固态存储设备上将操作系统加载到 RAM 中运行，整个过程并没有用户的介入。这种模

式是 Boot Loader 正常工作模式，因此在嵌入式产品发布时，Boot Loader 必须工作在这种模式下。

下载模式：在这种模式下，目标机上的 Boot Loader 将通过串口或网络等通信手段从开发主机下载内核映像和根文件系统等到 RAM 中，然后再被 Boot Loader 写到目标机上的 Flash 中，或者直接进行系统的引导。前一种功能通常是用于第一次烧写内核与根文件系统到 Flash 或者系统更新时使用；后者多用于开发人员在前期开发的过程中。工作于这种模式下的 Boot Loader 通常都会向它的终端用户提供一个简单的命令行接口。

4．Bootloader 体系结构

从操作系统的角度来看，Bootloader 的总目标就是正确地调用内核来执行。Bootloader 的启动过程分为是单阶段（Single Stage）或多阶段（Multi-Stage）。通常多阶段的 Bootloader 能提供更为复杂的功能以及更好的可移植性。从固态存储设备上启动的 Bootloader 大多都是两个阶段的启动过程，也即启动过程可以分为 stage 1 和 stage 2 两部分[3]。依赖于 CPU 体系结构的代码，通常都放在 stage1 中，采用汇编语言来实现，以达到短小精悍的目的。stage2 通常用 C 语言来实现，这样可以实现更复杂的功能，而且代码会具有更好的可读性和可移植性。

4.1. Stage1 的操作

1. 基本硬件的初始化，包括以下步骤：
   (1) 屏蔽所有中断；
   (2) 置 CPU 速度和时钟频率；
   (3) 初始化 RAM；
   (4) 初始化 LED 或 UART；
   (5) 为加载 stage2 准备 RAM 空间。
2 复制 stage2 到 RAM 空间；
3 设置堆栈指针 SP，为执行 C 语言代码做准备；
4 跳转到 stage2 的 C 语言入口点。

4.2. Stage2 的基本操作

1. 初始化本阶段要用到的硬件设备；
2. 检测系统的内存映射，将存储在 Flash 上的内核映像到预留的 RAM 空间之前，先确定这些预留的 RAM 空间哪些真正映射到了 RAM 地址单元；
3. 加载内核映像和根文件系统映像；
4. 设置内核启动参数，Linux2.4.x 以后的内核都期望以标记列表的形式传递参数，在嵌入式 Linux 系统中，通常由 Bootloader 设置的启动参数有：ATAG_CORE，ATAG_MEM，ATAG_RAMDISK，ATAG_INITRD 等；
5. 调用内核，即直接跳转到内核的第一条指令地址处执行。

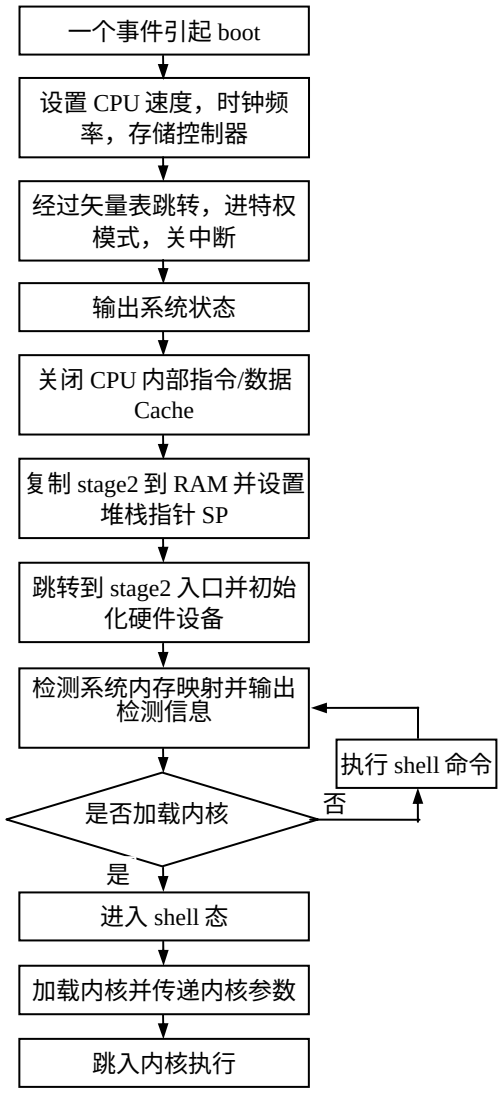

图 2．BootLoader 工作流程

4.3. Boot loader 工作流程

Bootloader 的工作流程如图 2 所示。

5．U-BOOT 主要特性

U-Boot 是德国 DENX 小组开发的用于多种嵌入式 CPU 的 Bootloader 引导程序, 支持 PowPC,ARM、MIPS、m68 K 等多种处理器平台,易于裁剪和调试。U2Boot 遵循 GPL (通用公共许可) 公约,完全开放源代码。U-BOOT 主要特性如表 1 所示。

表 1．U-BOOT 主要特性

功　　能	描　　述
系统加载	支持 NFS、串口、以太网、USB 挂载操作系统、根文件系统
内存操作	支持内存察看、修改、比较
设备驱动	支持串口、FLASH、外部 SDRAM、EEPROM、LCD、USB、PCI 等驱动
上电自检	自动检测 FLASH、SDRAM 选型及使用情况，CPU 选型
交互命令	通过设定和访问环境变量灵活配置系统各项参数，灵活升级

6．U-BOOT 源码分析

6.1. 硬件资源

AT91RM9200 是 Atmel 公司推出的一款采用 ARM920T 内核的高性能的嵌入式芯片，主要用于工业控制、性能要求比较高的场合。该芯片的主要特点有：主频为 180MHz，有内存管理单元，支持 LINUX 操作系统，外围总线接口支持 SDRAM，静态存储器、Burst Flash，有以太网（Ethernet MAC 10/100 Base-T），USB2.0 全速达到 12Mbit/s，SPI 接口，TWI 接口，4 个 USART 接口，3 个 SSC 接口等等。以 AT91RM9200 为核心的嵌入式系统板的配置为：

1. FLASH 存储器： SPANSION 公司生产的容量为 2M 的 S29GL064M；
2. SDRAM 存储器： HYUNDAI 公司生产的容量为 32M 的 HY57V281620；
3. 以太网接口芯片： DAVICOM 公司 DM9161；
4. 串口： 用于调试；
5. 晶振：外部 18MHz，CPU 主频 180M。

6.2. U-BOOT 的目录结构

U-BOOT 是一个通用的免费开放源代码的 boot 程序支持很多 CPU，表 2 列出了针对 AT91RM9200 系统的主要目录和文件。

表 2.U-BOOT 主要目录结构

路　　径	主要文件描述
/board/ at91rm9200dk	支持 AT91RM9200 目标板的子目录，at91rm9200dk.c，flash.c 分别为 SDRAM、FLASH 驱动
/cpu/ at91rm9200	存放支持的 cpu 类型，含串口、网口及中断初始化等文件，其中 start.S 文件完成对底层硬件的初始化
/common	存放了一些公共命令，与内核相关的两个文件是 cmd_boot.c，cmd_bootm.c
/drivers	存放各种外设接口的驱动
/fs	存入了支持的文件系统
/include	存放头文件的公共目录，其中 include/configs/at91rm9200dk.h 定义了所有和 at91rm9200 处理器目标板相关的资源配置参数，如波特率、引导参数、物理内存映射等

6.3. U-Boot 运行分析

U-Boot 的启动过程也主要分为两个阶段，即 stage1 和 stage2。其中 stage1 用汇编语言编写，在 /cpu/ at91rm9200/start.S 中实现；stage2 为 C 语言程序,用来加载操作系统内核，是由/lib_arm/board.c 中的 start-armboot() 函数来实现。

6.4. 系统启动流程

首先，在 board\ at91rm9200dk\ u-boot.lds 文件中指定 start 是整个程序的总入口：

```
.text      :
{
  cpu/at91rm9200/start.o(.text)
  *(.text)
}
```

其中，start 在 cpu\ at91rm9200\ start.S 文件中定义，入口代码如下：

```
.globl _start
_start:b       reset
ldrpc, _undefined_instruction /*arm 中断向量表
```

……

```
reset：/*系统复位后跳转到 reset
    mrs     r0,cpsr         /*基本硬件初始化，操作
    bic     r0,r0,#0x1f   /* CPRS，把 CPU 设为
    orr     r0,r0,#0x13  /* SVC32 模式

            /*初始化一些系统 Cache 寄存器和 RAM 以
            /*便从 RAM 运行程序
            /*把 BOOT 代码和数据重新定位到 RAM
    ……
copy_loop：
    ldmia r0!,{r3-r10} /*r0=程序在 flash 中入口地址
    stmia r!,{r3-r10}   /*r1=目标地址
    cmp r0,r2             /*r2=复制长度
    ble copy_loop

    ldr r0, _armboot_end   /*调整堆栈指针 sp 到堆栈
                           /*顶部
    add r0, r0, #CONFIG_STACKSIZE
    sub sp, r0, #12

    ldr pc,_start_armboot   /*跳转到 C 语言如函数
                           /*_start_armboot
```

C 语言入口函数定义在 lib_arm/board.c 中，其主要功能是调用一些初始化函数，实现大部分硬件设备初始化和硬件开发板的全局数据结构的初始化。

其中，start_armboot ()函数中调用 main_loop () 进入命令循环，接收来自用户从串口输入的命令或直接执行先定义的命令。

7．结束语

Bootloader 的开发是一个非常复杂的过程，其工作量相当于一个小型的操作系统。由于有了 U-BOOT 这样良好的源代码结构，以及广大程序开发者的技术支持，因而具有良好的扩展性。本文分析了基于 AT91RM9200 嵌入式系统的部分 U-BOOT 源代码，以制定符合自己要求的引导程序。

参考文献

[1] The Denx U-Boot and Linux Guide. Http://www.denx.de/twiki/bin/view/DULG/Mannual, 2004-08-12.

[2] 詹荣开．嵌入式系统 Bootloader 技术内幕[EB/OL].http://www-900.ibm.com/developmentworks/cn/linux/1-btloodex.2003-12.

[3] 陈渝，李明，杨晔. 源码开放的嵌入式系统软件分析与实践—基于 SkyEye 和 ARM 开发平台. 北京: 北京航空航空天大学出版社, 2004, 136-201

[4] 石教英等.计算机体系结构. 杭州: 浙江大学出版社,2003.