BootLoader 与 Linux 内核的参数传递

夏传凯，吴乃陵
东南大学电子科学与工程学院，南京(210096）
E-mail：xchk215@163.com

摘　要：本文介绍了嵌入式系统 BootLoader 与 Linux-2.6.19.2 内核参数传递的具体实现。主要内容包括：系统硬件平台的简介，参数链表的结构和创建，内核的接收。

关键词：BootLoader，AT91RM9200，Linux，Tagged list，内核参数传递

0. 引言

在嵌入式系统中，BootLoader 是用来初始化硬件，加载内核，传递参数。因为嵌入式系统的硬件环境各不相同，所以嵌入式系统的 BootLoader 也各不相同，其中比较通用的是 U-Boot，它支持不同的体系结构，如 ARM，PowerPC，X86，MIPS 等。本文着重介绍 BootLoader 与内核之间参数传递这一基本功能。本文的硬件平台是基于 AT91RM9200 处理器系统，软件平台是 Linux-2.6.19.2 内核。内核映像文件为 zImage。

1. 系统硬件平台简介

AT91RM9200 处理器，它是由 Atmel 公司基于 ARM920T 内核的微处理器，带有内存管理单元，CPU 时钟最高可达 240MHz,它具有丰富的标准接口，EBI 接口，内部集成了静态存储控制器（SMC），SDRAM 控制器，Burst Flash 控制器。有关处理器的说明请参考 AT91RM9200 的数据手册。本系统 SDRAM（64MB）地址为：0x20000000, NorFlash（8MB）的地址为：0x10000000[1]。系统硬件平台的原理图如下：

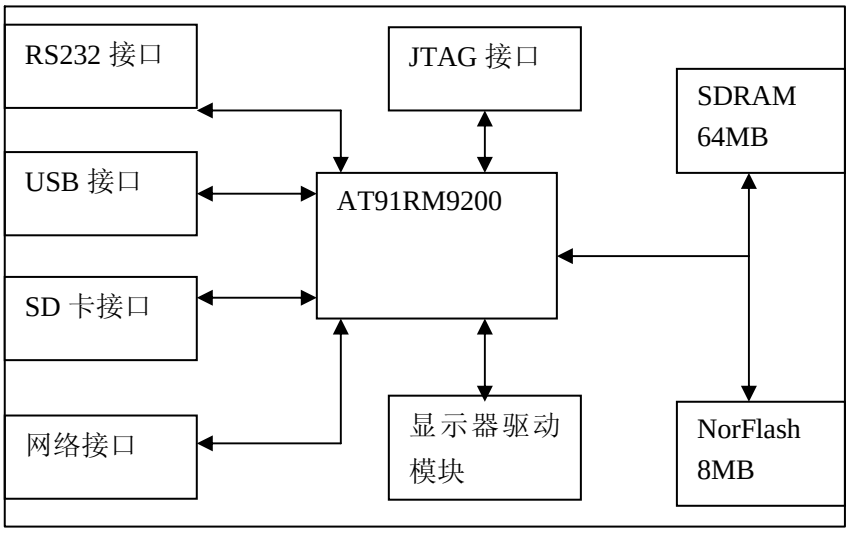

图 1　系统硬件平台原理图

2. BootLoader 设计和实现

内核源代码目录树下的 documentation/arm/booting[2]文档规定了基于 ARM 体系结构 BootLoader 的基本功能。本系统 BootLoader 除了完成这些基本的功能外，还结合自身硬件的特点加入了代码搬运等功能。

BootLoader 的流程是：系统上电复位后，首先从 NorFlash 开始运行（由处理器 BMS 引脚连接决定），因为处理器此时的 0 地址就是 NorFlash 的首地址(0x10000000)，BootLoader 就是被烧写在这个位置，AT91RM9200 处理器能够映射的地址范围只有 0x0000 0000—0x001f ffff。 BootLoader 执行的第一步就是将自身代码从 NorFlash 中搬运到处理器内部的 RAM 中（0x00200000），然后将 0 地址映射到内部 RAM,并且跳转到内部 RAM 的相应地址处继续执行。进入内部 RAM 后才进入真正的硬件初始化阶段，这个阶段初始化的各种控制器都是内核所必须的，包括：PMC, EBI, SMC, SDRAM, USART 等。接着就是创建内核参数链表(Tagged list)，创建完链表就是搬运事先烧写在 NorFlash 中的内核映像和根文件系统映像到 SDRAM，根据内核对 BootLoader 的基本要求关闭中断，MMU 和数据 Cache，并且配置 r0=0, r1=0x0000 00fb 或者 0x00000106(根据内核中 linux/arch/arm/tools/mach-types[2]规定的机器编号)，r2=0x20000100（BootLoader 传递给内核参数链表的物理地址），在 ARM 体系结构中，这个地址在同一种处理器的机器描述符（machine_desc）中都是默认的，所以在这里可以不指定。最后 BootLoader 直接跳转到 SDRAM 的内核处执行。

3. 内核参数链表

BootLoader 可以通过两种方法传递参数给内核，一种是旧的参数结构方式（parameter_struct)，主要是 2.6 之前的内核使用的方式。另外一种就是现在的 2.6 内核在用的参数链表 (tagged list) 方式。这些参数主要包括，系统的根设备标志，页面大小，内存的起始地址和大小，RAMDISK 的起始地址和大小，压缩的 RAMDISK 根文件系统的起始地址和大小，内核命令参数等[3][4][5]。

内核参数链表的格式和说明可以从内核源代码目录树中的 include/asm-arm/setup.h[2]中找到，参数链表必须以 ATAG_CORE 开始，以 ATAG_NONE 结束。这里的 ATAG_CORE，ATAG_NONE 是各个参数的标记，本身是一个 32 位值，例如：ATAG_CORE=0x54410001。其它的参数标记还包括： ATAG_MEM32， ATAG_INITRD， ATAG_RAMDISK，ATAG_COMDLINE 等。每个参数标记就代表一个参数结构体，由各个参数结构体构成了参数链表。参数结构体的定义如下：

```
struct tag {
        struct    tag_header    hdr;
        union {
                struct tag_core             core;
                struct tag_mem32          mem;
                struct tag_videotext      videotext;
                struct tag_ramdisk        ramdisk;
                struct tag_initrd           initrd;
                struct tag_serialnr         serialnr;
                struct tag_revision         revision;
                struct tag_videolfb         videolfb;
                struct tag_cmdline          cmdline;
                struct tag_acorn             acorn;
                struct tag_memclk         memclk;
              } u;
   };
```

参数结构体包括两个部分，一个是 tag_header 结构体,一个是 u 联合体。

tag_header 结构体的定义如下：

```
struct tag_header {
                    u32 size;
                    u32 tag;
                };
```

其中 size：表示整个 tag 结构体的大小(用字的个数来表示，而不是字节的个数)，等于 tag_header 的大小加上 u 联合体的大小，例如，参数结构体 ATAG_CORE 的 size=(sizeof(tag->tag_header)+sizeof(tag->u.core))>>2，一般通过函数 tag_size(struct * tag_xxx)来获得每个参数结构体的 size。其中 tag：表示整个 tag 结构体的标记，如：ATAG_CORE 等。

联合体 u 包括了所有可选择的内核参数类型，包括：tag_core, tag_mem32，tag_ramdisk 等。参数结构体之间的遍历是通过函数 tag_next(struct * tag)来实现的。本系统参数链表包括的结构体有：ATAG_CORE，ATAG_MEM，ATAG_RAMDISK，ATAG_INITRD32，ATAG_CMDLINE，ATAG_END。在整个参数链表中除了参数结构体 ATAG_CORE 和 ATAG_END 的位置固定以外，其他参数结构体的顺序是任意的。本 BootLoader 所传递的参数链表如下：第一个内核参数结构体，标记为 ATAG_CORE，参数类型为 tag_core。每个参数类型的定义请参考源代码文件。

tag_array 初始化为指向参数链表的第一个结构体的指针。

```
tag_array->hdr.tag=ATAG_CORE;
    tag_array->hdr.size=tag_size(tag_core);
    tag_array->u.core.flags=1;
    tag_array->u.core.pagesize=4096;
    tag_array->u.core.rootdev=0x00100000;
    tag_array=tag_next(tag_array);

    tag_array->hdr.tag=ATAG_MEM;
    tag_array->hdr.size=tag_size(tag_mem32);
    tag_array->u.mem.size=0x04000000;
    tag_array->u.mem.start=0x20000000;
    tag_array=tag_next(tag_array);
    ……
    tag_array->hdr.tag=ATAG_NONE;
    tag_array->hdr.size=0;
    tag_array=tag_next(tag_array);
```

最后将内核参数链表复制到内核默认的物理地址 0x20000100 处。这样参数链表就建好了。

4. 内核接收参数

下面从基于 ARM 体系结构的 zImage 映像启动来分析 Linux 内核是怎样接收 BootLoader 传递过来的内核参数，zImage 启动过程如下图所示。

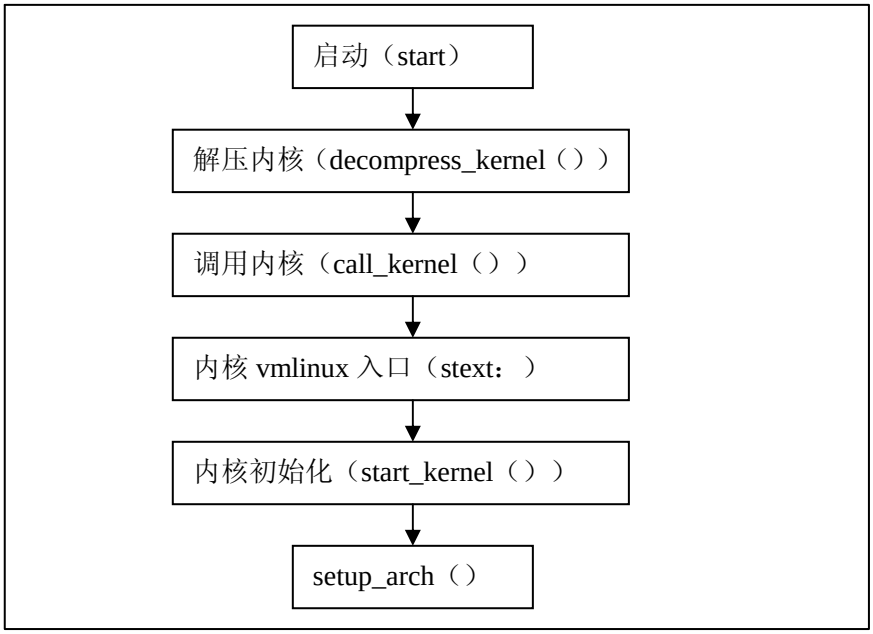

图 2　参数传递的路径[5]

在文件 arch/arm/boot/compressed/head.S[2]中 start 为 zImage 的起始点，部分代码如下：

```
start:
        mov r7, r1
        mov r8, r2
        …...
        mov r0, r4
        mov r3, r7
        bl    decompress_kernel
        b     call_kernel
call_kernel：
        ……
        mov r0, #0
        mov r1, r7
        mov r2, r8
        mov pc, r4
```

首先将 BootLoader 传递过来的 r1（机器编号）、r2（参数链表的物理地址）的值保存到 r7、r8 中，再将 r7 作为参数传递给解压函数 decompress_kernel（）。在解压函数中，再将 r7 传递给全局变量__machine_arch_type。在跳到内核（vmlinux）入口之前再将 r7，r8 还原到 r1，r2 中。

在文件 arch/arm/kernel/head.S[2]中，内核（vmlinux）入口的部分代码如下：

```
stext：
    mrc  p15, 0, r9, c0, c0
bl __lookup_processor_type
    ……
    bl    __lookup_machine_type
```

首先从处理器内部特殊寄存器（CP15）中获得 ARM 内核的类型，从处理器内核描述符（proc_info_list）表（__proc_info_begin—__proc_info_end）中查询有无此 ARM 内核的类型，

如果无就出错退出。处理器内核描述符定义在 include/asm-arm/procinfo.h[2]中，具体的函数实现在 arch/arm/mm/proc-xxx.S[2]中，在编译连接过程中将各种处理器内核描述符组合成表。接着从机器描述符（machine_desc）表（__mach_info_begin—__mach_info_end）中查询有无 r1 寄存器指定的机器编号，如果没有就出错退出。机器编号 mach_type_xxx 在 arch/arm/tools/mach-types[2]文件中说明，每个机器描述符中包括一个唯一的机器编号，机器描述符的定义在 include/asm-arm/mach/arch.h[2]中，具体实现在 arch/arm/mach-xxxx[2]文件夹中，在编译连接过程中将基于同一种处理器的不同机器描述符组合成表。例如，基于 AT91RM9200 处理器的各种机器描述符可以参考 arch/arm/mach-at91rm9200/board-xxx.c[2]，机器编号为 262 的机器描述符如下所示：

```
MACHINE_START(AT91RM9200DK, "Atmel AT91RM9200-DK")
/* Maintainer: SAN People/Atmel */
.phys_io  = AT91_BASE_SYS,
.io_pg_offst    = (AT91_VA_BASE_SYS >> 18) & 0xfffc,
.boot_params  = AT91_SDRAM_BASE + 0x100,
.timer            = &at91rm9200_timer,
.map_io         = dk_map_io,
.init_irq   = dk_init_irq,
.init_machine  = dk_board_init,
MACHINE_END
```

最后就是打开 MMU，并跳转到 init/main.c[2]的 start_kernel（）初始化系统。

在 init/main.c[2] 中，函数 start_kernel（）的部分代码如下：

```
{   ……
   setup_arch（）；
   ……}
```

在 arch/arm/kernel/setup.c[2]中，函数 setup_arch（）的部分代码如下：

```
 { ……
setup_processor();
mdesc=setup_machine(machine_arch_type);
……
parse_tags(tags);
……
 }
```

setup_processor（）函数从处理器内核描述符表中找到匹配的描述符，并初始化一些处理器变量。setup_machine（）用机器编号（在解压函数 decompress_kernel 中被赋值）作为参数返回机器描述符。从机器描述符中获得内核参数的物理地址，赋值给 tags 变量。然后调用 parse_tags（）函数分析内核参数链表，把各个参数值传递给全局变量。这样内核就收到了 BootLoader 传递的参数。

5. 参数传递的验证和测试

参数传递的结果可以通过内核启动的打印信息来验证。

```
Machine: Atmel AT91RM9200-DK
    ……
Kernel command line: console=ttyS0,115200 root=/dev/ram rw init=/linuxrc
```

……

Memory: 64MB = 64MB total

……

checking if image is initramfs...it isn't (no cpio magic); looks like an initrd

Freeing initrd memory: 1024K

……

RAMDISK: Compressed image found at block 0

6. 结束语

一个完备的 BootLoader 是一个很复杂的工程，本文所介绍的只是嵌入式系统的 BootLoaer 基本功能。任何一个 BootLoader 都离不开这个基本功能，内核只有接收这些参数才能正确地启动。目前，笔者的 BootLoader 已经能够稳定地运行在硬件平台上，为以后开发更复杂和完备的 BootLoader 打下了一定的基础，同时也为内核的移植和调试奠定了良好的基础。

参考文献

[1] 杜春雷. ARM 体系结构与编程[M].北京；清华大学出版社,2003.2

[2] Linux-2.6.19.2 kernel, www.kernel.org

[3] 李汉强,邱巍. 基于 Intel PXA26X 处理器的 BootLoader 的设计与实现[J]. 武汉理工大学学报（交通科学与工程版）2003,12:27-6

[4] 郑家玲,张云峰,孙荷琨.嵌入式系统的内核载入过程浅析[J].微型计算机应用，2002 年第 11 期

[5] 孙纪坤,张小全. 嵌入式 Linux 系统开发技术详解-基于 ARM[M].北京；人民邮电出版社,2006.

Parameters transferring from bootloader to linux

Xia Chuankai, Wu Nailing

Southeast University, Nanjing, Jiangsu (210096)

Abstract

This article introduce parameters transferring from bootloader of embedded system to linux-2.6.19.2 kernel and implement. It consists of simple introduction of hardware system, structure and implement of kernel tag lists and how to be received by kernel .

Keywords: BootLoader AT91RM9200 Linux Tagged list Kernel tagged list transferring

作者简介：

夏传凯，1980，男，硕士研究生；

吴乃陵，男，教授，硕士生导师，E-mail：wunailing@mail.edu.cn。