

# ARM-Linux 嵌入式系统 BootLoader 的配置与移植

孟 雷, 忽海娜

(许昌学院 计算机科学与技术学院, 河南 许昌 461000)

**摘 要:** 嵌入式系统 BootLoader 的配置及移植是解决嵌入式系统的启动、初始化、操作系统内核的固化和引导等问题的关键。分析了 ARM-Linux 嵌入式系统的组成, 介绍了 BootLoader 的概念及移植、修改需要注意的问题。并以 S3C44B0 处理器为例, 实现了其在 Linux 系统下的配置与移植, 通过引导程序 BootLoader 的装入, 成功解决了嵌入式系统的启动、初始化、操作系统内核的固化和引导等问题。

**关键词:** 嵌入式系统; 配置; 移植

**中图分类号:** TP333.5

**文献标识码:** A

**文章编号:** 1673-629X(2008)10-0204-03

## ARM-Linux Embedded System BootLoader Configuration with Transplantation

MENG Lei, HU Hai-na

(College of Computer Science & Tech., Xuchang University, Xuchang 461000, China)

**Abstract:** Embedded system configuration and transplantation is the key to resolve the questions such as the start of initialization, the operating system kernel and solidifying & inducing of operating system kernel. Mainly includes: analysis of ARM-Linux embedded system constitution, introduction of BootLoader concept, and some issues in transplantation and modifying. Taking S3C44B0 processor as an example, realizes the configuration and transplantation in the Linux system. By loading the inducing program BootLoader, resolve the start & initialization of embedded system.

**Key words:** embedded system; configure; transplant

## 0 引 言

32 位 ARM 嵌入式处理器具有高性能、低功耗的特性, 已被广泛应用于消费电子产品、无线通信和网络通信等领域。目前, 大批大型高科技企业以及为数众多的中小型企业, 甚至高等院校的实验室, 都将使用重点放在 Linux 系统上<sup>[1]</sup>。利用嵌入式 Linux 系统开发出来的大型移动通信设备、ATM 交换设备、IP 交换设备已经在市场上稳定运行, 甚至医药仪器和航空、航天及军事领域, 也得到了很好的使用。而嵌入式系统的启动引导技术是嵌入式系统开发的一个难点。系统启动引导的成功与否决定了应用程序的运行环境是否能正确构建, 即系统启动成功是系统正确运行的前提。

常用的嵌入式系统启动方法是先通过 JTAG(边界扫描测试技术)将嵌入式操作系统内核写进 Flash, 再由引导程序 BootLoader 完成嵌入式系统的启动引导

工作<sup>[2]</sup>。完成这个工作要分两个步骤进行, 即映像的下载, 软件以及硬件的调试。利用引导程序 BootLoader 的装入, 解决了嵌入式系统的启动、初始化、操作系统内核的固化和引导等问题。

## 1 Linux 嵌入式系统的系统组成

### 1.1 系统的硬件组成

嵌入式系统在设计上的特殊性决定了其组成上的特殊性。不过, 所谓嵌入式系统是个谱系, 所以实际上并不存在“标准”的嵌入式系统组成。当然, 每一个系统都有 CPU、存储器、输入输出这三大组成部分。这个基本结构是不会变的。典型的 ARM 嵌入式系统硬件平台一般包括一个以 ARM 为内核的处理器、存储器和必要的外部接口与设备。

文中的 ARM-Linux 嵌入式系统启动引导的实现是以三星公司的 S3C44B0 作为 ARM 芯片的<sup>[3]</sup>。

#### 1.1.1 嵌入式处理器芯片的特点

三星公司的 S3C44B0 作为 ARM 芯片处理器<sup>[4]</sup>具备如下特点:

收稿日期: 2008-01-10

基金项目: 河南省科技攻关计划项目(0524220009)

作者简介: 孟 雷(1981-)男, 助教, 研究方向为嵌入式系统、计算机网络。

- 1) 高性能、低功耗。
- 2) 支持以太网接口。
- 3) 丰富的中断控制器,以满足与 FPGA 接口的需要。
- 4) LCD 控制器。
- 5) 至少拥有 2 个 UART 接口(CPU 与 GPRS 模块接口为 UART)。
- 6) 拥有 12C 总线控制器。
- 7) 基于 ARM 的收视率终端设备,实现丰富的 I/O 接口,以方便与 FPGA 连接。
- 8) 为保证系统的可靠性,具有看门狗功能。

### 1.1.2 可行性分析

可以看出,这是一个功能比较齐全的 ARM7 位处理器,只需要介入少量外围电路,就可以满足各类嵌入式系统设备的开发需要。嵌入式仪器网络控制器需要独立运行,S3C44B0 自带的 LCD 控制器可以极大地满足用户交互的需求。同时,成熟的网络、USB 扩展方案与 FPGA 的扩展方案大大增强了系统的性能,由此可见,三星 ARM7 处理器 S3C44B0 完全可以满足各类嵌入式系统设备的设计要求。

## 1.2 系统的软件组成

嵌入式系统的软件平台由以下部分组成:系统引导程序、嵌入式操作系统内核、文件系统(后两者在此不作详细叙述,需要者可查阅这方面的资料)。

### 1.2.1 BootLoader 的概念

BootLoader 是系统加电运行的第一段代码<sup>[5]</sup>。简单地讲 BootLoader 就是在操作系统内核或用户应用程序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(有的 CPU 没有内存映射功能如 S3C44B0),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或用户应用程序准备好的环境。对于一个嵌入式系统来说,可能有的包括操作系统,有的小型系统也可以只包括应用程序,但是在这之前都需要 BootLoader 为它准备一个正确的环境。通常,BootLoader 是依赖于硬件而实现的,特别是在嵌入式领域,为嵌入式系统建立一个通用的 BootLoader 是很困难的。

### 1.2.2 BootLoader 的移植和修改

每种不同的 CPU 体系结构都有不同的 BootLoader。除了依赖于 CPU 的体系结构外,BootLoader 实际上也依赖于具体的嵌入式板级设备的配置,比如板卡的硬件地址分配,RAM 芯片的类型,其他外设的类型等。这也就是说,对于两块不同的嵌入式板而言,即使它们是基于同一种 CPU 而构建的,如果他们的硬件资源和配置不一致的话,要想让运行在一块板子上的

BootLoader 程序也能运行在另一块板子上,还是需要作一些必要的修改。

系统加电或复位后,所有的 CPU 通常都从 CPU 制造商预先安排的地址上取指令。比如,S3C44B0 再复位都从地址 0x00000000 取它的第一条指令。而嵌入式系统通常都有某种类型的固态存储设备(比如:ROM、EEPROM 或 FLASH 等)被安排这个起始地址上,因此在系统加电后,CPU 将首先执行 BootLoader 程序。也就是说对于基于 S3C44B0 的这套系统,我们的 BootLoader 是从 0 地址开始存放的,而这块起始地址需要采用可引导的固态存储设备如 FLASH。

## 2 BootLoader 配置与移植

### 2.1 Blob 简介与配置

Blob 是 Boot Loader Object 的缩写<sup>[6]</sup>,是一款功能强大的 BootLoader。可以用来简单地调试,也可以用来启动存放在 Flash 或 RAM 中的 Linux kernel。Blob 程序最大定义为 64kB,分成两个部分启动<sup>[7]</sup>。第一阶段的代码在 start.s 中定义,大小为 1kB,它包括从系统上电后在 0x00000000 地址开始执行的部分。这部分代码运行在 Flash 中,它包括对 S3C44B0 的一些寄存器的初始化和将 Blob 第二阶段代码从 Flash 拷贝到 SDRAM 中。除去第一阶段的 1kB 代码,剩下的部分都是第二阶段的代码。第二阶段的起始文件为 trampoline.s,被复制到 SDRAM 后,就从第一阶段跳到这个文件开始执行剩余部分代码。第二阶段最大为 63kB,单词 trampoline 词义为“蹦床”<sup>[8]</sup>,所以在这个程序中进行一些 BSS 段设置、堆栈的初始化等工作后,最后跳转到 main.c 进入 C 函数。从第二阶段跳转到 main()后,需要完成如下工作:外围硬件初始化(串口,usb 等);从 Flash 中加载 kernel 到 SDRAM;从 Flash 中加载 ramdisk 到 SDRAM;进入命令行模式或启动 kernel。

### 2.2 S3C44B0 的 BootLoader 移植过程

下面针对 S3C44B0 的 BootLoader—Blob 作为系统的 BootLoader 进行分析。Blob 程序启动流程如图 1 所示。

进行 Blob 移植之前,首先要修改 Start.S 文件:

- 1) 配置寄存器 SYSCFG 暂时关闭缓存,等 Blob 运行稳定后再开启提高性能。
- 2) 初始化 IO 寄存器。
- 3) 屏蔽中断。
- 4) 配置 PLLCON 寄存器决定系统的主频。
- 5) 调用 ledasm.S,在串口未初始化时 led 状态对于程序是否正常运行很重要。
- 6) 调用 memsetup - s3c44b0.S 中的 memsetup 进

行初始化存储器空间,初始化 SDRAM 刷新速率等在 ledasm.S 中,提供了 led 的驱动程序,这里做的和 led.c 中工作一样,只不过是初期调试阶段使用。

在 memsetup - s3c44WS 中,修改 MEMORY - CONFIG 中设置存储器相关的配置,并设定 SDRAM 刷新速度。

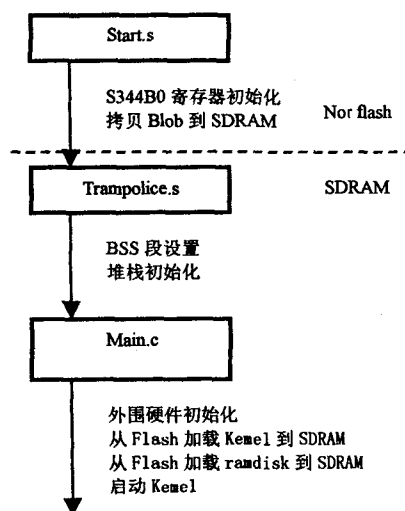


图 1 Blob 程序启动流程图

MEMORY - CONFIG:

```
long 0x11101002 /* 进行存储器的配置,SDRAM 刷新速度配置等 */
```

```
.long 0x20
```

```
.globl memsetup /* 定义全局标号,以便能被 Start.S 调用 */
```

```
memsetup:
```

```
ldrr0, =MEMORY - CONFIG /* 进行配置 */
```

```
ldmia r0, {r1 - r13}
```

```
ldrr0, =0x01c80000
```

```
stmia lr, {r1 - r13}
```

```
mov pc, lr /* 程序返回. /
```

Trampoline.S 不需要进行修改。

进入 Main 后,在结构体 blob\_status 中设定串口传输速度:

```
Blob_status.downloadSpeed = baud_115200;
```

```
Blob_status.terminalSpeed = baud_115200;
```

串口的初始化在 serial - s3c44b0.c 中的 s3c44b0\_serial\_init 中,一般只需要初始化下面四个寄存器串口就可以正常工作。如果不能工作,可能是系统不同,只需要按照下列公式计算出 divisor = (int)(MCLK / (bps x 16)) - 1 替换下面的 divisor 即可。其中 MCLK 为系统主频,bps 为波特率。/\* serial - s3c44b0.c 中 s3c44b0\_serial\_init() 函数初始化串口 0 部分

```
REG(ULCON0) = 0x0; /* 关闭 fifo */
```

```
REG(ULCON0) = 0x03; /* 设置数据位 8,无奇偶校验,一位停止位 */
```

```
REG(ULCON0) = 0x05; /* 脉冲中断,中断请求或查询模式 */
REG(UBRDIV0) = divisor; /* 设置波特率 */
```

至此,初级的移植已经完成,运行 ./configure - with - board = mba - 44b0 - with - linux - prefix = /path/to/linux - src 进行相关配置,然后 make 即可在 blob/src/blob 下得到 bin 格式的 blob,将其烧写到 Flash 即可。关于 Blob 第一部分和第二部分的链接脚本,可以在 tart - ld - script 和 rest - ld - script.in 中看到相关的链接地址。在 blob/src/blob/Makefile 中可以看到,两个部分分别以 blob - start 和 blob - rest 来编译命名,最终通过 dd 命令将它们组成一个完整的 blob 二进制文件。

### 3 结束语

这样整个基于 ARM - Linux 嵌入式系统 BootLoader 的配置与移植就基本上完成了,当然对于不同的系统,操作是略有不同的,嵌入式 BootLoader 对硬件的依赖性非常强,不同的 CPU 体系结构都有不同的 BootLoader。除了依赖 CPU 的体系结构外,BootLoader 实际上也依赖于具体的嵌入式板级设备的配置。也就是说使用同一款 CPU 的两块不同的嵌入式板子,它们之间的 BootLoader 也不是完全通用的。可以根据所要开发或使用的嵌入式系统模式,进行适当的调整,利用引导程序 BootLoader 的装入,解决了嵌入式系统的启动、初始化、操作系统内核的固化和引导等问题。

### 参考文献:

- [1] 刘晶晶. 基于 ARM - Linux 嵌入式系统引导程序的设计[J]. 微计算机信息, 2006, 16: 123 - 125.
- [2] 毛德操, 胡希明. 嵌入式系统[M]. 杭州: 浙江大学出版社, 2003: 640 - 656.
- [3] Yagbmour K. 构建嵌入式 LINUX[M]. 北京: 中国电力出版社, 2004: 232 - 259.
- [4] 马忠梅, 李善平. ARM9 & Linux 嵌入式系统教程[M]. 北京: 北京航空航天大学出版社, 2004: 196 - 200.
- [5] 马毅. 基于 linux 实时操作系统设计[J]. 计算机工程与应用, 2001(23): 290 - 297.
- [6] 翟鸿鸣. Linux 系统实时性能增强方法的研究[J]. 微机发展(现名: 计算机技术与发展), 2003, 13(s): 1 - 3.
- [7] 李驹光. ARM 应用系统开发详解[M]. 北京: 清华大学出版社, 2004: 302 - 311.
- [8] Kazmi M, Wiberg N. Scheduling Algorithms for HS - DSCH in a WCDMA Mixed Traffic Scenario[C]// Proceedings of the 14th IEEE 2003 International Symposium on Personal, Indoor and Mobile Radio Communications (Volume 2). [s. l.]: [s. n.], 2003.