U-Boot 在 S3C2440 上的分析与移植①

师　磊 (西安工程大学 电子信息学院 陕西 西安 710048)

摘　要： Bootloader (引导加载程序) 是嵌入式系统开发的重要环节,它使得操作系统和硬件平台联系起来,对嵌入式系统的后继软件开发十分重要。介绍了当前嵌入式开发中功能强大、稳定可靠的引导装载程序 U-Boot 的特点、移植的过程，并且实现了包括下载内核、yaffs 文件系统和启动 Linux 的功能,利用相应的命令下载内核、yaffs 文件系统和启动 Linux，证实了所移植的 U-Boot 的正确性。

关键词： 嵌入式系统；Bootloader；S3C2440；U-Boot

Analysis and Transplantation of U-boot on the S3C2440

SHI Lei

(College of Electronics and Information, Xi'an Polytechnic University, Xi'an 710048, China)

Abstract: Bootloader (boot loader) is an important part of linking up the operating system and hardware platform, and also important for embedded system of software development. This paper describes characteristics of a powerful, stable U-Boot program in the development of the current embedded systems and transplant process. It realizes functions like kernel and file system of yaffs download, and Linux boot. By using the corresponding commands to download the kernel, yaffs file system and boot Linux, it proves the U - Boot accurate.

Keywords: embedded system; Bootloader; S3C2440; U-boot

1　引言

Bootloader (引导加载程序) 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,我们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[1]。Bootloader 是紧密依赖于硬件而实现的,特别是在嵌入式系统中,不同体系结构的处理器需要不同 Bootloader,即使是基于同一种处理器构建的不同开发板,通常也需要进行不同的 Bootloader 移植工作。因此,在嵌入式系统中建立一个通用的 Bootloader 几乎是不可能的[2]。本文主要是针对最小硬件系统的 Bootloader 移植。

2　U-Boot特点与启动流程

2.1 U-Boot 的特点

U-Boot 是遵循 GPL 条款的开放源码项目，U-Boot 支持多种操作系统的引导，除了支持 Linux 系统的引导外，还支持 NetBSD、VxWorks、LynxOS 等嵌入式操作系统，并且支持 ARM、PowerPC、NIOS、XScale 等处理器。这正是 U-Boot 所具有的特点，即支持尽可能多的嵌入式处理器和嵌入式操作系统。有较高的稳定性和可靠性、丰富的设备驱动源码、高度灵活的功能设置。这些优点是致使它被广泛应用的原因，也是选择移植到 S3C2440 开发板上的原因。

2.2 U-Boot 的启动流程

U-Boot 的启动分为两个阶段：阶段 1 用汇编语言实现，阶段 2 用 c 语言实现。将依赖于处理器体系结构的汇编语言部分和通用的 c 语言部分分开，这样有利于 BootLoader 能够支持尽可能多的嵌入式处理器。

阶段 1 的主要内容包括：

(1) 硬件设备初始化 :屏蔽所有的中断 ;设置 CPU

① 收稿时间:2009-07-20;收到修改稿时间:2009-10-14

的速度和时钟频率；RAM 初始化等。

(2) 为加载 Bootloader 的阶段 2 准备 RAM 空间。

(3) 拷贝 Bootloader 的阶段 2 到 RAM 空间中。

(4) 设置好堆栈。

(5) 跳转到阶段 2 的 c 入口点。

阶段 2 主要内容包括：

(1) 初始化本阶段要使用到的硬件设备。

(2) 检测系统内存映射(memory map) 。

(3) 将 kernel 映像和根文件系统映像从 Flash 上读到 RAM 空间中。

(4) 为内核设置启动参数。

(5) 调用内核[2]。

3 U－Boot移植过程

3.1 移植环境

硬件环境： CPU：SAMSUN G S3C2440AL；SD-RAM 内存：板载 64MB SDRAM HY57V561620，地址范围:0x30000000-0x34000000；FLASH 存储器：板载 64MB Nand Flash K9F1208U0C,地址范围 :0x0000000 -0x3d09000 ;板载 1MB Nor Flash AM29LV160D。

软件环境：RedHat9.0；U-Boot-1.1.6；交叉编译工具：arm-linux-gcc-3.4.1 http：//www.handhelds.org(下载地址)。

本文采用的是友善之臂的 MINI2440 开发板，此开发板采用 400MHz 的 ARM920T 内核的处理器 S3C2440。它是一款应用于手持设备的低成本，高性价比的开发板，能够装载嵌入式 Linux 以及 Wince 操作系统。

3.2 移植步骤

BootLoader 移植除了依赖于 CPU 的体系结构外，还依赖于具体的嵌入式板级设备的配置，只需选择与自己板子最为相近的作为基础。该版本 U-Boot 并不支持本文使用的 ARM 微处理器 S3C2440,但是对于同一个系列的另一款微处理器 S3C2410 却有很完善的支持。两款芯片使用相同的 ARM 核,片内集成的功能以及寄存器的使用都很接近,所以 U-Boot 的移植工作在 S3C2410 和对应的开发板 SMDK2410 的基础上展开[3]。它与我所用的板子的硬件配置是最为相近的。值得注意的是两者的时钟频率是不同的。这将在下面论述。

(1) 在 board 目录下创建一新目录,命名为 jason2440 (根据自己喜好命名) 。

(2) 修改 U-Boot－1.1.6 目录下的 Makefile 文件，找到 SMDK2410_config : unconfig

@ $(MKCONF IG) $(@: _config =) arm arm 920t S3C2410 NULL S3c24x0，在其下仿照它加上如下代码：

jason2440_config : unconfig @$(MKCONF IG) $(@: _config =) arm arm920t S3C2440 NULL S3c24x0。

各项的意思如下:

arm：CPU 的架构(ARCH)

arm920t： CPU 的类型(CPU)，其对应于 cpu/arm920t 子目录。

jason2440： 开发板的型号(BOARD)，对应于 board/jason2440 目录。

NULL：开发者或经销商(vender),NULL 为没有。

S3c24x0：片上系统(SOC)。

(3) 将 smdk2410 目录下的文件拷入上述目录中，并将其中的 smdk2410.c 改名为 jason2440.c。同时还得修改 board/jason2440/Makefile 文件：COBJS := jason2440.o flash.o

(4) 在 include/configs 中将 smdk2410.h 复制一份在相同目录下，并改名为 jason2440.h。

(5) 修改 cpu/arm920t 中的 start.s 文件，u-boot 的阶段 1 代码通常放在此文件中。

修改中断禁止部分，加入 s3c2440 的中断设置：

ldr r1, =0x7fff //根据 s3c2440 手册，INTSUBMSK 有 15 位

ldr r0, =INTSUBMSK

str r1, [r0] //屏蔽中断服务

由于 u-boot1.1.6 并不支持 S3C2440，所以要设置 S3C2440 的时钟频率，使其工作在 400Mhz：

设置 FCLK、HCLK、PCLK 的关系为 1：4：8，根据 S3C2440 芯片手册可以得到如下的代码：

```
ldr r0, =CLKDIVN
mov r1, #0x5 //设置 FCLK、HCLK、PCLK 之间的比例关系
str r1, [r0]
mrc p15, 0, r1, c1, c0, 0
orr r1, r1, #0xc0000000 //设置 cpu 总线模式为
```

asynchronous 模式

```
mcr p15, 0, r1, c1, c0, 0
```

此外在#ifndef CONFIG_SKIP_LOWLEVEL_INIT

```
……
bl  cpu_init_crit
```

#endif 中是判断是否进行 cpu 包括内存在的初始化的，若想在内存中调试 U-Boot,则此时不能对内存进行初始化,否则就相当于把刚下载到内存的代码又删除[4],故应在 bl 后加上“ ne ”即 blne cpu_init_crit。

(6) 修改 board/jason2440 中的 lowlevel_init.s 文件，该文件是用来配置和初始化内存、flash 以及 sdram 的。根据自己硬件的设置来进行相应的配置。根据自己板子的硬件设置，修改如下：

```
#define B4_BWSCON????(DW16 + WAIT + UBLB)
#defined REFCNT  0x4f4
```

(7) 在 cpu/arm920t/s3c24x0 目录中加入 NAND Flash 读取函数 nand_flash.c.添加 S3C2440 nand 的 驱动函数，同时修改 nand_flash.c 中的 s3c24x0_inithw(void) 函数， 在 s3c24x0_nand_inithw(void)函数中定义 S3C2440 nand flash 的时钟和初始化 nand flash。

(8) 修改 cpu/speed.c 文件，设置 PCLK 时钟频率，使得串口能正确的输出信息。由于 S3C2410 和 S3C2440 的 MPLL、UPLL 计算公式不一样， 所以修改 get_PLLCLK 函数加入以下代码：

```
if (gd->bd->bi_arch_number = MACH_TYPE_jason2440)
return((CONFIG_SYS_CLK_FREQ * m * 2) / (p << s));
```

(9) 修改 u-boot 以支持烧写 yaffs 映像文件。yaffs 是一种专门针对 nand flash 设计的可读写型文件系统，它能针对嵌入式存储器的功能要求提供损耗平衡、掉电保护等。u-boot 没有提供对 yaffs 文件系统的读写支持，但由于该文件系统的读写过程与其他文件系统的读写类似，因此可以通过修改 u-boot 的 flash 读写命令，增加处理 00B 区域数据的功能，实现对 yaffs 文件系统的读写支持。在/common/cmd_nand.c 文件的 U_BOOT_CMD()宏函数中，仿照其他参数使用说明添加以下内容：

```
“nand read[.yaffs] -addr off|partition size\n”
“ nand write[.yaffs] -addr off|partition size –read/write size' bytes starting\n”
```

同时在 do_nand 函数中的 else if (s != NULL && !strcmp(s, ".raw"))前加入代码实现对 nand write.yaffs 命令的支持。在 include/nand.h 的 struct nand_write_options 函数中增加 skip-firstblk 成员，在/nand/nand_util.c 中修改 nand_write_opts 函数，增加对 skipfirstblk 成员的支持。skipfirstblk 表示烧写时跳过第一个可用的逻辑块，这是由 yaffs 文件系统的特性决定的。

(10)为引导 Linux 内核,在 board/jason2440.c 中修改开发板类型代码：

```
/* arch number of S3C2440 -Board */
gd->bd->bi_arch_number = MACH_TYPE_jason2440 ;
#endif
```

(11) 编写 cmd_main.c 文件，加入到/common 目录中。设计主函数以实现功能菜单的选择。为了实现包括下载内核、yaffs 文件系统和启动 Linux， 这里需要用到 strcpy 函数和 run_command 函数。strcpy 函数在/imclude/asm-mips/string.h 中，形式为 extern __inline__ char *strcpy(char *__dest, __const__ char *__src)，含义是把 src 所指的由 null 结束的字符串复制到 dest 所指的数组中；run_command 函数在/common/main.c 中定义，这个函数主要是对用户输入的命令进行语法分析，从中获取命令，参数等信息，并查找一张系统保存的命令表，找到该命令对应的处理函数。并调用它来处理这个命令。此命令表在 include/command.h 中定义。

4 实验结果

编译 U-Boot 所用的交叉编译工具是 arm-linux-gcc-3.4.1，编译之前需要注释掉/cpu /arm920t/config.mk 文件中的-msoft-float,否则会出现软浮点的问题。

在 u-boot 目录下输入: make jason2440_config

Make

生成五个文件: u-boot 、u-boot.bin 、u-boot.srec 、u-boot .map 、System.map 。通过 JTAG 将二进制文件 u-boot.bin 烧入 Flash 中即可,拔掉 JTAG 接头并复位目标板,超级终端输出如图 1 所示：

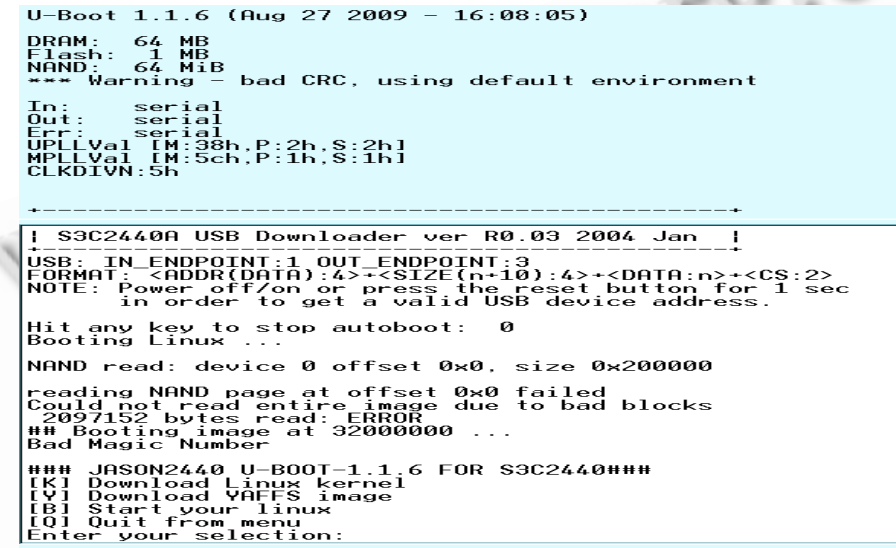

图 1 u-boot 运行结果

5 结论

Boot Loader是操作系统和硬件之间的桥梁,为操作系统的启动提供了必要的条件和参数。在移植过程中,除了要熟悉 U-Boot 的特点和流程外, 还要对相应的硬件平台有足够的认识。实验结果表明 U-boot1.1.6 移植到目标板的工作已经完成,并且能稳定的运行在目标板上,同时能够支持加载 Linux 内核、yaffs 文件系统和启动 Linux,为后续嵌入式系统软件的开发提供了良好的环境。

参考文献

1 孙琼.嵌入式 Linux 应用程序开发详解.北京:人民邮电出版社, 2006.

2 于云松,由德凯,孙其芳.基于 S3C2440 的 Bootloader 的设计与实现.甘肃科技, 2008,(8):24－26.

3 李军,张华春.U-Boot 及 Linux2.6 在 S3C2440A 平台上的移植方法.电子器件, 2008,(10):1667－1670.

4 王淑贞,姚 铭,周结华. U - Boot在S3c2410上的移植.微计算机应用, 2008,(4):95－99.