第30卷 第4期
2007年8月

电子器件
Chinese Journal Of Electron Devices

Vol.30 No.4
Aug.2007

Porting Method of U-Boot in S3C2440 Board

ZHANG Hui[1,2], ZHANG Hua-chun[1]

(1. Institute of Electronics, The Chinese Academy of Sciences, Beijing 100080, China;
2. Gradnate School of The Chinese Academy of Sciences, Beijing 100039, China)

Abstract: Bootloader U-Boot has enough functions and comprehensive applications, but when porting it to the ARM microprocessor S3C2440A is a relatively complex work. We briefly introduced many general Bootloaders, gathered up the main characters of U-Boot, analyzed the process of U-Boot in-depth, clearly explained the system memory distribution and the hardware resource configuration in our Embedded System board based on the CPU SAMSUNG S3C2440A, particularly introduced the porting method, porting process and main porting feature of U-Boot in the board. In the end, we get the results in the test experiments. Currently, U-Boot could achieve to the designed functions and runs steadily, establishing required foundation for farther porting Linux OS.

Key words: embedded system; porting; modification; bootloader; U-Boot; S3C2440A

EEACC: 1265

U-Boot 在 S3C2440 上的移植方法

张 徽[1,2], 张华春[1]

(1. 中国科学院电子学研究所, 北京 100080;
2. 中国科学院研究生院, 北京 100039)

摘 要: Bootloader U-Boot 功能齐全、应用广泛但移植到 ARM 微处理器 S3C2440A 上相对比较复杂. 简介了常见的 Bootloader, 归纳了 U-Boot 的主要特征, 分析了其运行过程, 介绍了系统存储空间分布和基于 S3C2440A 微处理器为核心自主开发的嵌入式系统板硬件资源配置, 给出了 U-Boot 在嵌入式系统板上的移植方法、移植过程和移植要点. 最后对实验结果进行了测试. 目前, U-Boot 可以完成设计的功能并能够稳定的运行, 为下一步移植 Linux 操作系统奠定了必需的基础.

关键词: 嵌入式系统; 移植; 修改; Bootloader; U-Boot; S3C2440A

中图分类号: TP368.1 **文献标识码**: A **文章编号**: 1005-9490(2007)04-1423-04

Bootloader 是严重依赖于硬件而实现的. 不同体系结构的处理器需要不同的 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 Bootloader 移植工作[1].

1 Bootloader 简介

Bootloader(引导加载程序)主要完成初始化硬件设备、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, 以便为调用操作系统内核准备好正确的环境[1]. 目前流行的可以引导加载 Linux 的 Bootloader 包括: 支持 x86 体系结构的 LILO, GRUB, ROLO, Loadlin, Etherboot, LinuxBIOS; 支持 ARM 体系结构的 Compaq 的 bootldr, Blob, U-Boot, vivi, RedBoot; 支持 MIPS 体系结构的 PMON 和支持 m68k 体系结构的 sh-boot[2].

U-Boot 是德国 DENX 小组开发的用于多种嵌入式 CPU 的 Bootloader 程序[3], 在支持 ARM 体系结构的 Bootloader 中间应用最为广泛, 目前版本是

收稿日期: 2006-08-24

作者简介: 张 徽(1982-), 男, 硕士研究生, 从事嵌入式系统研究, zhanghui-82@yeah.net.

1.1.4. Das U-Boot-universal bootloader 被认为是功能最多、最具弹性以及开发最积极的开放源码 Bootloader，表 1 给出了它的主要特征[4].

表 1 U-Boot 的主要特征

功能	描 述
系统加载	支持 NFB、串口、以太网、USB 挂载操作系统、根文件系统
内存操作	支持内存查看、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

2 U-Boot 运行过程分析

Bootloader 启动大多数都分为两个阶段. 第一阶段主要包含依赖于 CPU 的体系结构硬件初始化代码，通常都用汇编语言来实现. 第二阶段通常用 C 语言完成，以便实现更复杂的功能，也使程序有更好的可读性和可移植性[1]. 同时，对于整个嵌入式系统板的存储器空间分布有清晰的认识，有助于 U-Boot 的源代码修改和移植工作.

2.1 U-Boot 启动流程[1,4]

U-Boot 第一阶段主要任务有：① 本地硬件设备初始化(屏蔽所有中断、关闭处理器内部指令/数据 cache 等；② 为第二阶段准备 RAM 空间；③ 如果从固态存储介质中启动，则复制 Bootloader 的第二阶段代码到 RAM；④ 设置堆栈；跳转到第二阶段 C 程序入口点.

第二阶段主要任务有：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；③ 将内核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.

2.2 系统存储器空间分布

本文中 FLASH 存储空间分配[5-6]如图 1 所示.

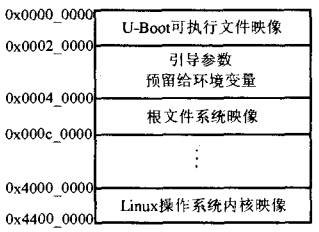

图 1 FLASH 存储空间分布

系统上电执行时，U-Boot 首先将自身搬运到 SDRAM 地址 0x33f80000，然后将 Linux 操作系统内核映像搬运到 SDRAM 地址 0x33000000 处，如果选择执行操作系统，系统将跳转到该处执行，SDRAM 中间存储空间分配[5-6]如图 2 所示.

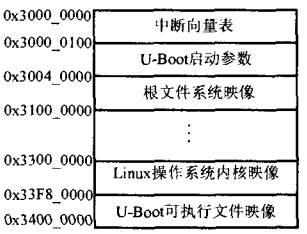

图 2 SDRAM 存储空间分配

3 U-Boot 移植操作

最新发布的 1.1.4 版本 U-Boot 并不支持本文所使用的微处理器 S3C2440，但是对于同一个系列的 S3C2410 却有很完善的支持. 本文 U-Boot 的移植工作在微处理器 S3C2410 和对应的系统板 smdk2410 的基础上展开.

3.1 系统板硬件资源配置

在自主开发的系统板上的主要硬件资源配置如下表 2 所示.

表 2 系统板硬件资源配置

硬件	描 述
CPU	S3C2440A
存储器	NOR FLASH 2M HY29LV160 NAND FLASH 64M K9F1208 SDRAM 64M HY57V561620
串口/JTAG 口	3 个 RS-485 口/1 个
晶振	12 MHz 内部锁相倍频 CPU 主频 120 MHz

3.2 建立交叉编译环境

首先要在宿主机(一般为 PC)上安装标准 Linux 操作系统，如 Red Hat Linux，然后安装交叉编译工具链即可建立交叉编译环境. 本文作者主机端(PC)开发平台选用 Linux 操作系统，使用的交叉编译工具为 arm-elf-tools-200303 14. sh[7].

3.3 移植操作内容

U-Boot 移植操作，实际上就是根据嵌入式系统硬件资源，对相关的文件进行修改.

3.3.1 移植文件操作

移植 U-Boot 到新的嵌入式系统板上包括两个层面的移植，第一层面是针对 CPU 的移植，第二层面是针对 BOARD 的移植，同时需要移植相关的头文件[3]. 与 BOARD 相关的文件位于 board/smdk2410 目录下，复制 smdk2410 目录并修改文件 lowlevel_init. S，flash. c，smdk2410. c. lowlevel_init. S 文件是完成系统板外部存储空间的配置；smdk2410. c 是通用 I/O 口的初始化，flash. c 是 FLASH 芯片的驱动程序. 与 CPU 相关的文件位于

cpu/arm920t 目录及其子目录 s3c24x0 下，文件 start.S 主要完成底层硬件初始化和代码搬运任务. 头文件也分为 CPU 和 BOARD 两个层次，与 BOARD 相关的头文件主要是 include/configs/smdk2410.h 和 include/flash.h，二者完成系统板参数的基本配置. 与 CPU 相关的头文件是 include/S3c2410.h 和 S3c24x0.h. 二者主要完成 CPU 内部寄存器和中断向量表设置.

除以上需要修改的文件外，公共代码(/common 目录下的文件)、网络传输代码(/net 目录)、驱动程序(/drivers 目录)3 部分根据不同的移植要求修改. 一般情况下，对 U-Boot 的功能进行扩充或添加不支持的设备时，需要修改相应的程序.

3.3.2　具体移植操作

(1) 头文件的修改

需要修改的头文件包括 S3c24x0.h，S3c2410.h，smdk2410.h 和 flash.h 涉及头文件修改的细节较多，在这里并不全部列举，重点是说明移植方法.

① 头文件 S3c24x0.h 的修改：该文件涉及的是 CPU 内部寄存器相对位置安排和寄存器的名称，按照 S3C2440 数据手册进行修改，举例如下

在/* LCD CONTROLLER (see manual chapter 15) */位置增加和 S3C2440 相关的代码[6]

```
#ifdef CONFIG_S3C2440
    S3C24X0_REG32   LCDINTPND;
    S3C24X0_REG32   LCDSRCPND;
    S3C24X0_REG32   LCDINTMSK;
    S3C24X0_REG32   TCONSEL;
#endif
```

② 头文件 S3c2410.h 的修改：将 S3c2410.h 复制为 S3C2440.h，该文件的主要内容是设置寄存器的地址，按照 S3C2440 数据手册中寄存器的地址进行修改，例如：

将

```
#define S3C2410_NAND_BASE0x4E000000
```

修改为

```
#define S3C2440_NAND_BASE0x4E000000
```

③ 头文件 smdk2410.h 的修改：将 smdk2410.h 文件复制为 smdk2440.h. 该头文件主要完成系统板的各种基本参数的配置，具体改动如下：

修改系统板参数为：

```
#define  CONFIG_S3C2440  1
#define  CONFIG_SMDK2440  1
#define  CFG_PROMPT  "SMDK2440# "
```

同时由于 U-Boot 不支持系统板上使用的 Flash HY29LV160，Flash 相关部分修改举例如下：

将

```
#define CONFIG_AMD_LV400  1
```

替换成

```
#define CONFIG_HYNIX_LV160  1
```

④ 头文件 flash.h 的修改：按照 HY29LV160 数据手册，增加 FLASH 生产 ID 数据，举例如下：

```
#define HYNIX_ID_LV160B  0x22492249
/* HY29LV160BT ID (16 M, bottom ) */
```

(2) 与 CPU 目标代码相关的修改

位于目录 cpu/arm920t/下的 start.S 文件是整个 Bootloader 程序的入口点，在移植过程中，不需要做实质性的改动，只需要增加 S3C2440 相关的定义即可，具体修改举例如下.

增加

CPU S3C2440 定义 defined(CONFIG_S3C2440).

关于 cpu/arm920t/s3c2x0/目录中间的文件也只需按照 S3C2440 数据手册做简单的修改即可.

(3) 与 BOARD 相关的目标代码的修改

复制目录 board/smdk2410/并更改名称为 smdk2440，依次修改该目录下的三个文件 smdk2410.c，lowlevel_init.S，flash.c 中的内容.

① 文件 smdk2410.c 的修改：将 smdk2410.c 重命名为 smdk2440.c，该文件主要是通用 I/O 口的初始化，修改举例如下：

引用的头文件由 s3c2410.h 更换为 S3C2440.h.

将设置系统时钟的部分宏常量修改为[8]

```
#define FCLK_SPEED 1
#elif FCLK_SPEED==1
#define M_MDIV  0x38
#define M_PDIV  0x2
#define M_SDIV  0x1
#endif
```

在函数 int board_init(void)中间添加

```
gpio->GPJCON = 0x01555555;
gpio->GPJUP = 0x000001FFF;
```

② 文件 lowlevel_init.S 的修改：在仿真阶段需要将下面两句程序注释掉，否则 U-Boot 会从 FLASH 中间寻找 SDRAM 配置数据，在 U-Boot 可执行映像文件烧写到 Flash 中间后，可以用这种方法配置 SDRAM.

```
ldr  r1, _TEXT_BASE
sub r0, r0, r1
```

③ 文件 flash.c 的修改：由于 Flash HY29LV160 符合 JEDEC 标准，所以文件内容并不需要做太大的改动，添加一些 Flash 信息即可.

(4) Makefile 文件的修改

① U-Boot 根目录下 Makefile 文件的修改：在 U-Boot-1.1.4/Makefile 中 ARM92xT Systerms 注释下面加入以下两行：

```
S3C2440_config:unconfig
@./mkconfig $(@:_config=) arm arm920t smdk2440 NULL s3c24x0
```

其中 arm 是 CPU 的种类，arm920t 是 ARM CPU 对应的代码目录，smdk2440 是自己系统板对应的目录. 同时，交叉编译工具安装在：/usr/local/bin 目录下，所以把 CROSS_COMPILE 设置成相应的路径：CROSS_COMPILE= /usr/local/bin/arm-elf-

② cpu/arm920t/目录下 Makefile 文件的修改：该目录下 config.mk 文件中间编译选项-mabi=apcs-gnu，在编译过程中会报错，U-Boot 的 mailinglist 给出的解决办法是：

将

```
PLATform_CPPFLAGS +=
$(call cc-option,-mapcs-32,-mabi=apcs-gnu)
```

修改成：

```
PLATform_CPPFLAGS +=
$(callcc-option,-mapcs-32,
$(call cc-option,-mabi=apcs-gnu),)
```

③ board/smdk2440 目录下 Makefile 文件的修改：由于原来文件夹下面的 smdk2410.c 重命名为 smdk2440.c，所以需要把 Makefile 文件中的目标文件 smdk2410.o 替换成 smdk2440.o. 简单地说，Makefile 文件描述了目标文件之间的依赖关系，以及指定编译过程中使用的工具[1].

4 目标文件的生成和测试

4.1 目标文件的生成

以管理员身份登陆到 Linux 操作系统，在 U-Boot 目录下依次运行以下命令[9]：

```
make clobber  /* 删除错误的 depend 文件 */
make clean
make smdk2440_config
make
```

之后会生成五个文件：

U-Boot，U-Boot.bin，U-Boot.srec，

U-Boot.map，System.map.

4.2 生成目标代码的测试

利用编制好的 Flash 烧写程序，通过 JTAG 将二进制文件 U-Boot.bin 烧入 Flash 的零地址. 烧写成功后，拔掉 JTAG 接头并复位系统板. 从超级终端输出如下信息：

```
U-Boot 1.1.4(Aug 21 2006-10:55:28)
U-Boot Code: 33F80000->33F9A0800
BSS->33FA4950
RAM Configuration:
Bank#0: 30000000 64 MB
Flash:  2 MB
In:     serial
Out:    serial
Err:    serial
SMDK2440 -
```

串口输出的以上信息表明，CPU 和串口已正常工作. U-Boot 提供的命令 flinfo 和 mtest 以及 md，mw 可以测试 Flash 和 SDRAM. 经过测试，可以正确地读出 Flash 信息及读写 RAM，表明 Flash 和 SDRAM 已正确初始化. U-Boot 提供的命令 setenv，printenv 和 saveenv 分别可以设置、打印输出和保存环境变量. 上述测试结果表明 U-Boot 移植成功并且可以在系统板上稳定的运行.

5 结论

本文研究了 U-Boot 在基于 S3C2440 系统板上的移植方法及具体的软件修改方法. 理解 Bootloader 工作机理和明确硬件资源配置是进行移植工作的前提. 本文移植的 Bootloader U-Boot 目前能稳定地运行在自主开发的嵌入式系统板上，实现了串口与 PC 通信，查看、修改内存，引导 Linux 操作系统等功能，为后续工作奠定了基础，希望本文能为做相关工作的开发者提供参考.

参考文献：

[1] 孙天泽，袁文菊，张海峰. 嵌入式设计及 Linux 驱动开发指南：基于 ARM9 处理器[M]. 北京：电子工业出版社，2005. 80-84，95-98.

[2] Karim Yaghmour. 构建嵌入式 LINUX 系统[M]. 北京：中国电力出版社，2004. 255-260.

[3] Current Versions. DENX software engineering [EB/OL]. http://www.denx.de/wiki/DULG/Manual. 2006，7.

[4] 吴平，曹晓琳，李 波等. 基于 ARM 核的 S3C4510B 启动代码的研究与应用[J]. 电子器件. 2004，27(2)：2-3.

[5] 王东，李哲英. U-Boot 在 S3C44B0 上的移植方法[J]. 北京交通大学学报. 2005，29(2)：2.

[6] 宋凯，甘岚，严丽平. U-Boot 在 S3C2410 上的移植分析[J]. 华东交通大学学报. 2005，22(5)：2-3.

[7] 龙 坤，陈亚军. 基于 μClinux-ARM 的嵌入式系统的设计[J]. 电子器件.. 2005，28(2)：3.

[8] S3C2440A 32Bit CMOS Microcontroller USER'S MAN-UAL. Samsung Electronics [EB/OL]. http://www.sam-sung.com. 2005，7.

[9] Configurations. DENX software engineering [EB/OL]. http://www.denx.de/wiki/DULG/Manual . 2006，7.