文章编号:1007－757X(2005)07－0048－03

U－Boot在S3C2410上的移植

曹程远

摘　要:bootloader是嵌入式系统中必不可少的用以完成系统的启动加载任务的一段软件代码。U－boot作为bootloader的集大成者,现已得到了广泛的应用。本文通过将u－boot移植到基于s3c2410的开发板上详细地介绍了u－boot的运行原理和移植过程。

关键词:bootloader;u－boot;s3c2410;移植

中图分类号:TP315;TP368　　**文献标识码**:B

引言

Bootloader是操作系统内核运行之前运行的一段小程序。主要用于初始化硬件设备,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。Bootloader与cpu的体系结构有关,不同的cpu体系结构都有不同的bootloader。同时bootloader还依赖于具体的嵌入式板级设备的配置。也就是说,对于两块不同的嵌入式板而言,即使基于同一种cpu构建,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,通常也都需要修改bootloader的源程序。本文就是将运行在目标板smdk2410上的bootloader(u－boot)移植到自己的开发板上。

1　U－Boot介绍

U－Boot是德国DENX小组的开发用于多种嵌入式CPU的开放源代码bootloader程序,目前版本是1.1.2。U－Boot是在ppcboot以及armboot的基础上发展而来。现已非常成熟和稳定,已经在许多嵌入式系统开发过程中被采用。目前支持的目标操作系统包括OpenBSD,NetBSD,FreeBSD,Linux,SVR4,Esix,Solaris,Trix,Sco,Dell,NCR,VxWorks,Lynxos,pSOS,QNX,RTEMS和ARTOS。就目前来看,U－Boot对PowerPC系列处理器支持最丰富,同时还支持MIPS,x86,ARM,Nios,XScale等诸多常用系列处理器。对Linux的支持最完善,是嵌入式Linux Bootloader的最佳选择。

由于本文要移植的开发板是基于s3c2410的开发板,在U－Boot中已有移植成功的smdk2410开发板,故以smdk2410开发板上运行的u－boot为参考针对自己的开发板进行移植。在U－Boot中有相对应的文件可以直接拿来修改使用。

开发板主要配置有:

◆CPU:S3C2410
◆FLASH:16M,INTEL TE28F128J3C－150
◆SDRAM:64M SAMSUNG K4S561632D－TC75
◆一个四线RS－232接口
◆ 一个10M/100M自适应以太网接口
◆ 一个TFT LCD接口
◆ 一个USB接口
◆ JTAG接口
◆ 运行状态指示LED灯

2　u－boot主要目录

◆ board－该目录下一般为针对特定板子的一些初始化和操作代码,如flash,doc等,ld文件也放在这里;

◆ cpu－该目录下是针对特定处理器的初始化和操作代码,启动代码.S文件也在这里;

◆ common－此目录存放独立于处理器体系结构的通用代码,如内存大小探测与故障检测,且该目录下main.c可以看作是linux中的sh,负责接受用户输入并送给相应的处理函数执行。

◆ driver－此目录下放的是各种驱动,如以太网驱动,LCD屏驱动等;

◆ doc－u－boot的说明文档;

◆ examples－目录下放的是可在u－boot下运行的例子,相当于linux中应用程序;

◆ fs－目录下是文件系统;

◆ include－目录下存放各种头文件和配置文件;

◆ lib—xxx－处理器体系相关的文件,如lib—arm目录就包含ARM体系结构相关的文件;

◆ net－与网络功能相关的文件目录,如bootp,nfs,tftp;

◆ post－上电自检文件目录;

◆ rtc－RTC驱动程序;

作者简介:曹程远,长沙理工大学汽车与机电工程学院,长沙　410076

◆ tools－目录下的代码都是可供使用的"工具".用于创建u－boot,bin镜像文件。因为是在宿主机上跑的,因此与前面不同,使用gcc编译;

3 u－boot运行过程

程序流程如图1

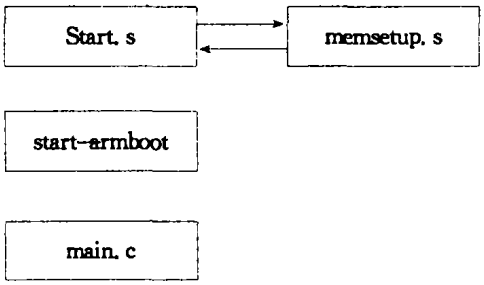

图1 u－boot程序流程图

u－boot大致分两个阶段,第一阶段依赖于cpu体系结构,由汇编语言实现,这部分代码主要包括cpu/arm920t目录下的start. s文件,被放在flash中;第二阶段用c语言实现,主要包括lib—arm目录下board. c文件中的函数start—armboot及common目录下main. c文件中的main—loop函数。

移植前需要对存储器的地址空间进行了解。存储器空间的定义在/include/configs/smdk2410. h中。存储器地址空间的分配见图2和图3。其中图2为在FLASH中的存储器空间分布,图3为启动后在SDRAM中的存储器空间分布。

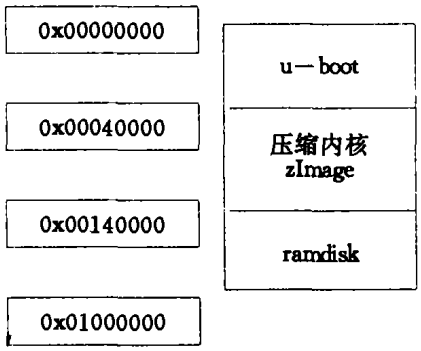

图2 FLASH布局

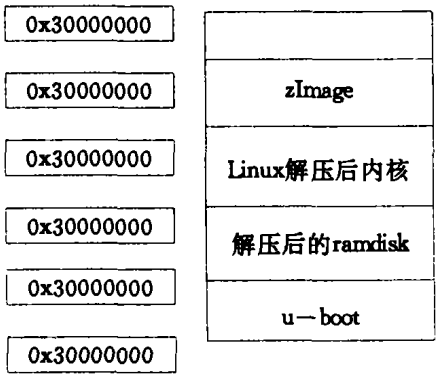

图3 SDRAM布局

如图所2示,flash存储器空间大小为16M,共分配给了u－boot,压缩内核zImage以及根文件ramdisk。SDRAM大小为64M. U－Boot第二阶段的代码就放在SDRAM空间最顶端的1M处,起始地址为0x33F00000,这是一种值得推荐的方法。

系统上电复位后从0x00000000地址处开始执行代码start. s(即第一阶段),这一部分代码运行在flash中。主要完成必要的寄存器设置,中断向量表,堆栈初始化等,并将第二阶段拷贝到SDRAM地址0x33F00000处,然后从第二阶段(即跳转到start—armboot函数)开始执行,调用各种init函数,完成第二阶段要使用的硬件设备初始化工作(主要是板级初始化),最后跳转到main—loop函数(属第二阶段)中,负责接受用户命令,然后将其分发给相应的处理函数。

4 移植主要修改文件

移植u－boot到开发板上只需要修改和硬件相关的代码即可。这首先就联想到cpu目录下的启动代码,另外参考u－boot/readme文件可知其他还需要修改的主要文件有:Makefile文件,和include目录下的目标板. h头文件(smdk2410. h),board目录下的目标板. c文件(smdk2410. c),flash. c文件,u－boot. lds链接文件,以及cpu目录下的串口驱动文件。具体修改如下:

① cpu/arm920t目录下

◆ start. s启动代码。缺省的处理器时钟值为120MHz,将其改为202. 8MHz,并初始化系统时钟。添加如下代码:

```
#define LOCKTIME 0x4c000000
#define MPLLCON 0x4c000014
#define MDIV—200 oxa1
#define PDIV—200 ox3
#define SDIV—200 Ox1
#define vMPLLCON—200 ((MDIV—200<<12)m
(PDIV—200<<4)m (SDIV—200))
.align 4
mpll—200mhz: .long vMPLLCON—200
ldr r1,=LOCKTIME
mvn r2,#0xff000000
str r2,[r1]
ldr r1,=MPLLCON
ldr r2,mpll—200mhz
str r2,[r1]
```

② board/smdk2410目录下,(因为参考的是smdk2410目标板)

◆ smdk2410. c文件。这个文件主要是SDRAM的驱动程序,主要完成SDRAM的UPM表设置,上电初始化。暂时不改。

◆ flash. c文件。Flash的驱动程序就在此文件中。将MAIN—SECT—SIZE由0x10000改为0x20000(因为INTEL TE28F128J3C－150共128个扇区,每个扇区都是128k字节

大小。)根据TE28F128J3C－150可知,这是一150ns的Flash,因为Flash相对于DSP来说是慢速设备,编程时,对Flash的访问需要有足够的延时等待。故把flash时序设为最慢。把开发板设成202.8MHZ(缺省时钟为120MHz),并且工作在异步模式。修改memsetup.s文件,也可以在调用memsetup.s之前的start.s文件中添加。添加如下:

设置flash时序并工作在异步模式:

```
Mrc p15,0,r1,c1,c0,0
Orr r1,r1,#0xc0000000
Mcr p15,0,r1,c1,c0,0
```

◆ memsetup.s文件。此文件基本不用修改,但为了以后能用uboot的GO命令执行修改过的用loadb或tftp下载的u－boot,在标记符"0:"处加上下面5句:

```
mov r3,pc
ldr r4,=0x3FFF0000
and r3,r3,r4 //以上三句得到实际启动的内存地址
add r0,r0,r3 //用GO命令调试uboot时,启动地址在SDRAM中
add r2,r2,r3 //把初始化内存信息的地址,加上实际启动的地址
```

◆ config.mk文件。此文件用于设置程序链接的起始地址。这里给u－boot预留了1M空间,故修改33F80000为33F00000。

◆ u－boot.lds文件。

③ include/configs目录下smdk2410.h文件。此文件是smdk2410目标板头文件,大多数寄存器参数是在这一文件中设置完成的。文件没有配置TE28F128J3C－150,故在此添加配置如下:

```
#define CONFIG—INTEL—28F128J3C 1
#if CONFIG—INTEL—28F128J3C
#define PHYS—FLASH—SIZE 0x01000000 /* 16MB */
#define CFG—MAX—FLASH—SECT (128) /* max number of sectors on one chip */
#define CFG—ENV—ADDR (CFG—FLASH—BASE + 0x00020000) /* addr of environment */
# define CHIP—MAIN—SECT—SIZE 0x20000 /* 128k */
#endif
```

5 u－boot移植过程

① 在宿主机上建立交叉编译开发环境

首先在linux机上建立交叉编译开发环境。可按照DENX和MontaVista提供的完整的开发工具集进行,也可自己建立交叉编译环境。本文采用的源码包版本为:linux－2.4.21.tar.bz2,patch－2.4.21.bz2,binutils－2.14.tar.gz,

gcc－core－2.95.3.tar.gz,gcc－g++－2.95.3.tar.gz,glibc－2.2.4.tar.gz。

② 修改cpu/arm920t目录中的文件内容,主要包含cpu.c,start.s,interrupts.c以及serial.c,speed.c等文件,其中start.s在上文中已修改。

③ 在board目录下创建自己的目标板(开发板)目录mys3c2410,在目录下创建mys3c2410.c,flash.c,memsetup.s以及Makefile,u－boot.lds,config.mk文件,将smdk2410目录下修改过的文件拷贝到相对应的文件中。

④ 在include/configs目录下创建mys3c2410.h,将smdk2410.h的修改过的内容复制其中。

⑤ 打开u－boot目录下Makefile文件,加入如下两行:

```
mys3c2410—config:unconfig
@./mkconfig $(@:—config=) arm arm920t mys3c2410
```

⑥ 编译。运行命令:

```
#make mys3c2410—config
#make
```

编译成功。生成基本的u－boot.

⑦ 烧写。把编译成的u－boot.bin烧写到到开发板flash中。

至此移植u－boot过程结束。

6 结束语

本文介绍了u－boot在s3c2410上的移植,目前已经可以在自己的开发板上稳定的运行,并且能够烧写到flash。

参考文献:

[1]宋国军,张侃谕,林学龙 嵌入式系统中u－boot基本特点及其移植方法[J].单片机与嵌入式系统应用,2004(10).

[2]杜春雷,ARM体系结构与编程[M].北京:清华大学出版社,2004.

[3]s3c2410处理器手册[DB/OL]. http://www.hhcn.com/chinese/embedlinux－res.html.

[4]http://cvs.sourceforge.net/viewcvs.py/u－boot/u－boot/.

[5]http://www.denx.de/twiki/bin/view/DULG/Manual.

(收稿日期:2005－01－15)