基于 S5PV210 的 U-boot 分析与移植①

熊星星[1], 何月顺[2]

[1](东华理工大学 机械与电子工程学院, 南昌 330013)
[2](东华理工大学 软件学院, 南昌 330013)

摘 要: 介绍嵌入式系统开发中功能强大、稳定可靠的引导装载程序(Bootloader)在基于 CotexA8 微处理器的 S5PV210 上的移植. 详细叙述包括启动流程、移植分析与移植步骤, 并且实现包括内核装载、ysffs2 根文件系统加载等功能. 最后利用写入 U-Boot 启动参数自行通过 TFTP 方式装载内核和 NFS 方式加载根文件系统, 最终启动嵌入式 Linux 系统, 证实了所移植的 U-Boot 在处理器 S5PV210 上引导启动 Linux 的可行性.

关键词: 嵌入式 Linux 系统; Bootloader; S5PV210; 移植; U-Boot

U-Boot Analysis and Transplantation Based on S5PV210

XIONG Xing-Xing[1] , HE Yue-Shun[2]

[1](School of Machinery and Electronic Engineering, East China Institute of Technology, Nanchang 330013, China)
[2](School of Software, East China Institute of Technology, Nanchang 330013, China)

Abstract: This paper describes powerful, reliable Bootloader in the development of Embedded Systems transplant on the S5PV210 microprocessor based on Cotex-A8. It narrates the boot process, analysis of transplantation and procedure of transplantation in detail and implements loading the kernel and ysffs2 root filesystem. Finally, it uses the way of written U-Boot boot parameters to automatic load the kernel via TFTP and mount the root file system via NFS. It ultimately starts embedded linux system and confirms the feasibility of the transplanted U-Boot booting Linux System on the S5PV210.

Key words: embedded linux system; bootloader; S5PV210; transplantation; U-Boot

Bootloader 是操作系统内核启动前的一小段引导程序. 通过这段程序, 可以初始化硬件设备、建立内存空间映射, 进而将系统的软硬件环境引导进入一个合适的工作状态, 为调用操作系统的内核准备好正确的环境[1]. Bootloader 依赖于硬件和应用环境, 因此为嵌入式系统建立一个通用、标准的 Bootloader 是难度的. 一般需根据硬件对 Bootloader 进行相应的修改, 以满足实际开发需要[2]. 本文对 U-boot 源码进行分析并将其移植到基于 Cotex-A8 的微处理器 S5PV210 上, 并给出了测试结果.

1 U-Boot 启动流程

S5PV210 上电复位后首先执行的是 IROM 中已固化的启动代码——BL0. 在 BL0 中, 检测 OM Pins, 判断何种设备启动, 同时从启动设备拷贝 BL1 到 IRAM 中并对其进行校验, 校验通过跳转 BL1 执行, BL1 完成基本的硬件初始化, 同时拷贝 BL2 到 IRAM 中并对其进行校验, 校验后跳转 BL2. BL2 完成全面的硬件初始化, 之后将 OS 代码拷贝到 SDRAM 中, 并跳转到 OS 中执行并完成启动引导. 启动流程如图 1 所示.

SP5V210 的 U-boot 启动过程分为三个阶段, 参照 S5PV210 手册[3]进行简单介绍. 三个阶段如下:

初始阶段(BL0):

主要初始化基本的系统功能: 初始化系统时钟、部分特殊控制器和启动设备等硬件; 依据 OM Pin 选择何种存储器启动; 加载 BL1—从外部启动设备如外扩存储器(nand/sd)等拷贝 BL1 映像至 Internal SRAM(0xD002000 开始的地址)处; 根据 BL1 头文件信

① 收稿时间:2014-04-22;收到修改稿时间:2014-06-03

息, 对 BL1 映像进行完整性校验, 通过校验转入 BL1 执行.

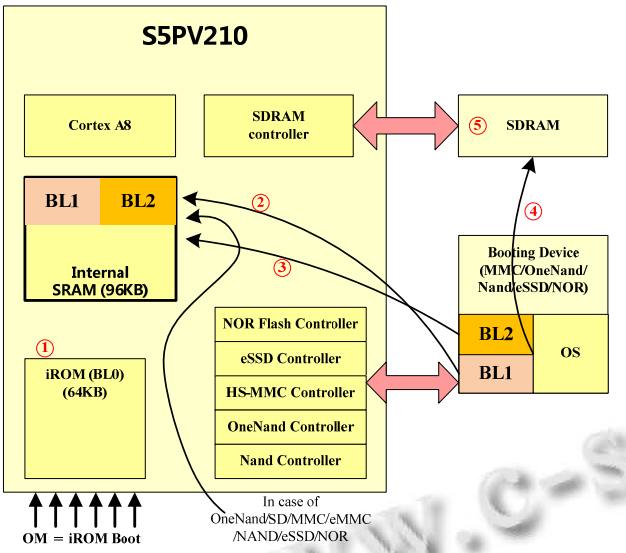

图 1 S5PV210 启动流程

第一阶段(BL1):

该阶段代码最大能不超过 16KB, 其中包含 0xD002000~0xD0020010 的 16 字 BL 头文件信息(BL1 的校验信息和 BL1 尺寸)和实际的 BL1 代码.

主要完成: 关闭看门狗、初始化 icache、建立临时堆栈、初始化 DRAM 控制器等基本硬件; 加载 BL2——从外部启动设备拷贝 BL2 映像至 Internal SRAM; 对 BL2 映像进行完整性校验, 通过校验转入 BL2 执行.

第二阶段(BL2):

该阶段代码实际上包含了大部分的 BL1 代码, 大小近乎是完整的 U-boot 代码.

主要完成: 串口、TZPC (TrustZone Protection Controller)等其他硬件的初始化; 加载内核——从启动设备拷贝内核至 DRAM, 将 CPU 的控制权交给系统 OS.

启动设备:

根据 S5PV210 IROM 文档[4], S5PV210 支持的启动方式有多种, 常用的使用 SD/MMC 和 NAND 方式, 本文中使用 SD/MMC 的方式引导启动 U-boot.

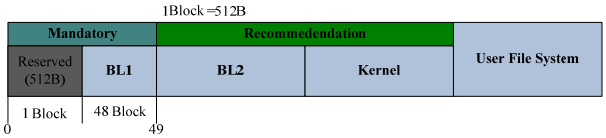

图 2 SD/MMC 设备引导区块图

SD/MMC 的 1Block 大小为 512B. Block0 为保留块, Block1-Block48 (24KB) 中前 16 Block(8KB)区域存放 BL1, BL1 的起始块是 Block1; 余下的 32 Block(16K)区域存放环境变量; 紧接着存放 BL2 文件, BL2 起始块是 Block49. SD/MMC 设备引导区块图如图 2 所示.

2 U-Boot移植

2.1 移植环境

硬件环境: Tiny210V2 开发板——CPU: SAMSUNG S5PV210(Internal ROM:64KB, Internal SRAM:96KB); DRAM 内存: 板载 512MB DRAM K4T1G084QF, 地址范围 0x2000_0000-0x3FFF_FFFF; FLASH 存储器: 板载 2GB NAND Flash K9GAG08U0F, 地址范围 0xB000_0000-0xBFFF_FFFF.

软件环境: Ubuntu 12.04 LTS; U-Boot-2013.1.rc[5].

交叉编译工具: arm-none-linux-gnueabi-gcc 4.3.3.

本文采用的 Tiny210V2 开发板, 是基于 Cotex-A8 ARM 内核的处理器 S5PV210, 该处理器主频最高可以达到 1GHz. 其特点是是低成本, 高性价比, 能够良好运行嵌入式 Linux、Wince 以及当前流行的 Android 操作系统.

2.2 U-Boot 移植分析与步骤

Bootloader 移植除了依赖于嵌入式板级设备的配置外, 还依赖于 CPU 的体系结构, 可以选择与目标板较为相近的平台作为移植基础.

U-boot 源码并不支持基于 S5PV210 的开发板, 我们需寻求支持与 S5PV210 相类似的处理器. 了解到 S5PV210 与 S5PC1XX 处理器都是基于 CortexTM-A8 的三星处理器, 同时 U-boot 支持基于 S5PC100 的 SMDKC100 开发板. 虽然两者较为相近, 但需修改及添加的代码量依旧比较多, 因此, 我们参考了开源公司 Linago 开发的支持 min210_linago for u-boot[6].

2.2.1 移植分析

首先分析 Makefile 文件, 该文件是 U-boot 编译的规则文件, 依照该文件编译生成 U-boot 可执行文件; 而后根据 Bootloader 执行过程分阶段进行移植分析, 即对 BL1 和 BL2 阶段进行详细分析.

Makefile 分析

首先打开顶层目录下的 Makefile 文件, 找到 all, all 目标依赖于$(ALL -y),

```
all:  $(ALL-y) $(SUBDIR_EXAMPLES)
```

ALL –y 依赖于$(obj)u-boot.bin, 文件 u-boot.bin 是完整的 U-boot 文件. 执行 make 命令, 则最终生成 u-boot.bin 文件.

```
ALL-y += $(obj)u-boot.srec $(obj)u-boot.bin $(obj)System.map
```

通过查找$(obj)u-boot.bin, 发现其依赖于 u-boot 文件, 依据编译工具和编译选项, 生成 u-boot.bin 文件

```
$(obj)u-boot.bin: $(obj)u-boot
 $(OBJCOPY) ${OBJCFLAGS} –O binary $< $@
 $(BOARD_SIZE_CHECK)
```

$(obj)u-boot 文件是编译器链接得到的文件, 依赖于库文件、中间文件、脚本文件及工具等.

```
$(obj)u-boot: depend \
 $(SUBDIR_TOOLS) $(OBJS) $(LIBBOARD) -
 $(LIBS) $(LDSCRIPT) $(obj)u-boot.lds
 $(GEN_UBOOT)
```

分析 Makefile 文件, 发现 u-boot-spl.bin 文件依赖于$(CONFIG_SPL), 即需定义 CONFIG_SPL 宏.

```
ALL-$(CONFIG_SPL) += $(obj)spl/u-boot-spl.bin
```

然后分析 spl/Makefile, 文件一开始就添加并导出 CONFIG_SPL_BUILD 宏. 所以生成 BL1 文件是依赖于该宏的, 同时可以确定编译 BL1 与 BL2 的代码范围. spl/目录即为生成 BL1 的关键目录.

```
CONFIG_SPL_BUILD := y
export  CONFIG_SPL_BUILD
```

S5PV210 是三星芯片, 定义了 CONFIG_SAMSUNG 宏, 则根据以下的命令生成 $(BOARD)-spl.bin, $(BOARD)=tiny210, 所以最终生成 spl/tiny210-spl.bin.

```
ifdef CONFIG_SAMSUNG
$(obj)$(BOARD)-spl.bin: $(obj)u-boot-spl.bin
                $(OBJTREE)/tools/mk$(SOC)spl  \
                $(obj)u-boot-spl.bin
                $(obj)$(BOARD)- spl.bin
endif
```

依据语句$(OBJTREE)/tools/mk$(SOC)spl , 在目录 tools/下查找到 mks5pc1xxspl($(SOC)= s5pc1xx)工具, 该工具是由 mks5pc1xxspl.c 文件编译生成而来, 通过分析分析源码, 其作用是在文件 u-boot-spl.bin 前面添加 BL1 头文件信息(16 个字长度)输出得到 $(BOARD)-spl.bin, 即 tiny210-spl.bin.

BL1 详细分析

以 start.S 为主线, 即分析该汇编文件. 首先定义中断向量表以及一些全局变量的初始化, 运行至标签 reset 处, bl save_boot_pargams 调用 save_boot_pargams 子函数(由于该子函数中是空指令, 因此并没有保存实际的启动参数). 而后设置 CPU 进入 SVC32 模式(安全运行模式), 之后通过宏 CONFIG_SKIP_LOWLEVEL_INIT, 判定是否跳过低水平初始化. S5PV210 BL1 流程如图 3 所示.

低水平初始化包含两个过程, 主要体现在下列两条执行语句中:

1. bl cpu_init_cp15 跳转至该子函数(禁用 L1 I/D、MMU 以及 caches);

2. bl cpu_init_crit 跳转至该子函数(函数内容 bl lowlevel_init, 跳转至标签 lowlevel_init(标签位于 board/tiny210/lowlevel_init.S)处执行 lowlevel_init.

lowlevel_init 功能介绍如下:

低水平初始化主要是完成读取复位标志, 如果是唤醒状态跳过硬件初始化; 禁用看门狗; 初始化 sram 和 PMIC; 判定 U-boot 是否在 SDRAM 中运行, 如果是则要跳过系统时钟以及 DRAM 控制器的初始化(/board/Samsung/tiny210/mem_setup.S 初始化内存), 否则需要完成时钟和 DRAM 控制器的初始化; 其他硬件的初始化(串口、TZPC、Flash 等); 判定是处于唤醒状态, 如果不是则关闭 ABB, 然后返回, 否则直接从内核启动. 不论低水平初始化执行与否, 都会执行到标签 call_board_init_f 处, 该代码段主要是完成设置堆栈和执行 bl board_init_f. 通过我们查看 spl/u-boot-spl.map 文件, 查询到 board_init_f 位于 board/ Samsung/tiny210/libti-ny210.o 中, 这个文件是依赖于当前目录下的 Makefile 生成的. 通过分析 Makefile 文件, 发现在宏 CONFIG_SPL_BUILD 中添加 COBJS+= mmc_boot.o, 而后在 board/Samsung /tiny210/mmc_boot.c 中查询到函数 board_init_f(), 该函数是在 C 语言环境中, 因此在调用之前需设置堆栈等.

函数 board_init_f()两个功能如下:

1) 复制 BL2 至 IRAM(执行 copy_u-boot_to_ram());

2) 程序跳转至 BL2 处执行

BL1 代码的作用: 进行一些简单的初始化, 而后将 BL2 复制到 RAM 中, 最后跳转到 BL2 处开始执行.

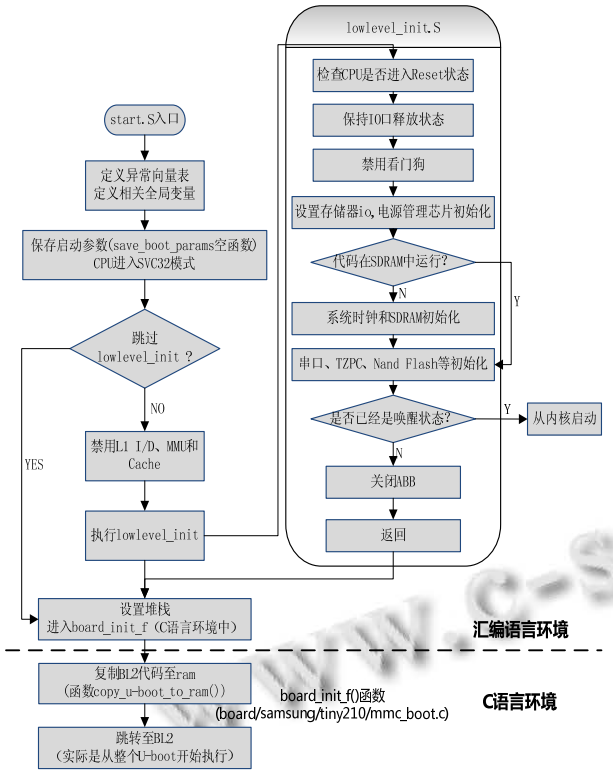

图 3　S5PV210 BL1 流程图

BL2 详细分析

S5PV210 BL2 阶段的代码实际上是完整的 U-boot 代码, 也是从 start.S 开始分析. 有部分过程与 BL1 重复, 就不再重复分析. 不同的是在 start.S 执行的 bl board_init_f 的函数不同, 我们通过查看 u-boot.map 文件, 查询到 board_init_f 标签位于 arch/arm/lib/libarm.o 中, 该文件是依赖当前目录下的 Makefile 文件生成的, 通过分析 Makefile 文件, 未定义 CONFIG_SPL_BUILD 宏的处添加了 COBJS += board.o, 通过 arch/arm/lib/board.c, 发现 board_init_f() 函数位于该文件中.

函数 board_init_f()(位于 arch/arm/lib/board.c)功能: 分配 gd 数据结构体地址, 并对其初始化; 对基本硬件初始化(初始化定时器、环境参数、波特率设置、串口通信、控制台、配置内存等).

函数 board_init_f()最后会跳转到 relocate_code(代码重定位)这个处理过程需要汇编语言支持, 执行该函数会跳转至 start.S 的 ENTRY(relocate_code)重定位代码入口处.

ENTRY(relocate_code)段代码完成的功能是完成堆栈的建立、U-boot 代码复制、数据段搬移、清 bss、跳转至 board_init_r()(C 语言环境).

函数 board_init_r()(位于 arch/arm/lib/board.c)功能: 进一步完成硬件的初始化(初始化 Flash、环境参数、PCI、控制台、以太网和初始化并使能中断等.)

在函数 board_init_r()最后进入一个循环函数 main_loop(), 该函数就相当于 U-boot 的 main()函数.

函数 main_loop()大致的执行过程是, 进入 Bootdelay 延时计数, 如果退出延时计数, 则直接进入 U-boot 命令行操作; 如果延时计数超时, 则根据设置的 U-boot 的启动参数自动引导操作系统内核, U-boot 交出 CPU 控制权给操作系统内核, 由内核引导操作系统运行. S5PV210 BL2 流程图 如图 4 所示.

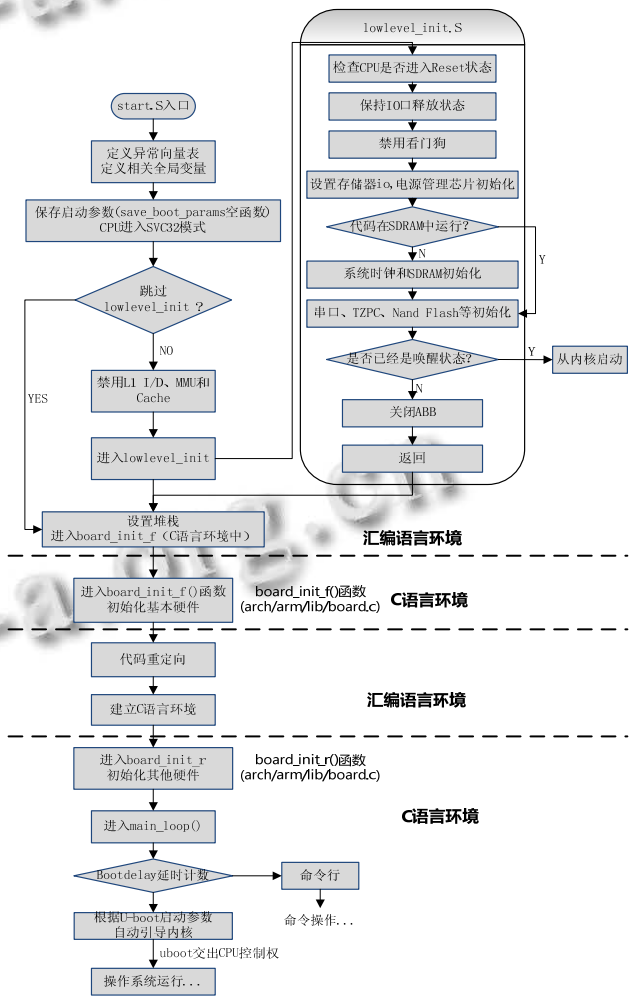

图 4　S5PV210 BL2 流程图

2.2.2 移植步骤

(1) 修改 U-boot-2013.01-rc 目录下 board.cfg 文件中找到如下行:

smdkc100 arm armv7 smdkc100 samsung s5pc1xx

该行代码, 在其上面仿照它加上如下一行代码:

tiny210 arm armv7 tiny210 samsung s5pc1xx

根据 board.cfg 文件开始处, 大致各项意思如下:

tiny210: 目标(TARGET)

arm: CPU 的架构(ARCH)

armv7: CPU 的类型(CPU), 其对应于 arch/cpu/armv7 子目录

tiny210: 开发板的型号(BOARD), 对应于 board/samsung/tiny210 目录

Samsung : 开发板的供应商(Vendor),

s5pc1xx: 片上系统(SoC), 由于 S5PV210 的芯片与 S5pc1xx 系列采用的都是 Cotex-A8, 所以我们参考 SMDKC110 U-boot 的代码.

NULL: (Options)该项该开发平台为空

(2) CPU 级目录

在文件 arch/arm/include/asm/arch-s5pc1xx/gpio.h 加入 s5p_gpio_part_max()函数.

arch/arm/include/arm/arch-s5pc1xx/mmc.h 修改代码比较多, 直接从 linago 代码同等目录下复制替换掉该文件.

arch/arm/include/arm/arch-s5pc1xx/hardware.h 文件添加至同等目录下.

(3) board 级目录

在 board/samsung 目录下创建一个板级目录 tiny210. 将 linago 的 board/samsung/tiny210 目录下的文件复制至该 tiny210 目录中. 在该目录下修改 tiny210.c 文件, 在 checkboard()函数中将开发板名称修改为 printf("\nBoard: TINY210V2 \n");

(4) include 级目录

将 linago 代码中 include/configs 目录下的 tiny210.h 复制至同等目录下, 同时修改下面宏 CONFIG_IDENT_STRING " for TINY210V2"、宏 CONFIG_IPADDR 和 CONFIG_SERVERIP 分别 192.168.1.230(开发板 IP 地址)和 192.168.1.220(远程服务器 IP 地址).

同时将 include/目录下的 s5pc110.h 和 s5pc11x.h 复制到相同目录下.

(5) drivers 级目录

drivers/mmc/ 目录下的 Makefile 中将 $(CONFIG_S5P_SDHCI) += s5p_sdhci.o 行的 s5p_sdhci.o 改成 s5p_mmc.o, 并且复制 linago 目录同等目录下 s5p_mmc.c 至该目录下.

drivers/mtd/nand/ 目录下的 Makefile 中添加 COBJS-$(CONFIG_NAND_S5PC1XX) += s5pc1xx_nand.o, 并且复制 linago 同等目录下 s5pc1xx_nand.c 至该目录下.

最后直接从 linago 的 drivers/usb/目录下的 gadget/和 host/两目录复制到该同等目录下.

(6) 修改 Makefile 和/spl/Makefile 两文件

Makefile 中将 CROSS_COMPILE ? = 改为 CROSS_COMPILE = arm-none-linux-gnueabi-(系统编译器), 并且将上一行的 ifeq ($(HOSTARCH),$(ARCH)) 改为 ifneg ($(HOSTARCH),$(ARCH)), 因为编译 ARCH(arm)与 HOSTARCH(X86)的架构是不同的. 同时将 $(obj)tools/mk{smdk5250,}spl 改为 $(obj)tools/mk{$(SOC),$(BOAR-D)}spl.

同时在/spl/Makefile 中 LIB -$ 最后添加以下两行:

LIBS-$(CONFIG_SPL_USB_SUPPORT) += drivers/usb/gadget/libusb_gadget.o

LIBS-$(CONFIG_SPL_MUSB_SUPPORT) += drivers/usb/musb/libusb_musb.o

最后在 $(OBJTREE)/tools/mk$(BOARD)spl?\ 该行处, 将 BOARD 改成 SOC.

3 测试结果

编译 U-Boot 所用的交叉编译工具是 arm-none-linux-gnueabi-gcc-4.3.3, 在 U-boot 目录下输入: make tiny210_config 命令进行配置

配置完成后输入: make 命令进行编译

主要生成文件: 在 u-boot-2013-01-rc/目录下包含 u-boot 和 u-boot.bin 等. 在目录 u-boot-2013-01-rc/spl 目录下包含 u-boot-spl、u-boot-spl.bin 和 tiny210-spl.bin 等文件. 测试需要 tiny210-spl.bin(BL1) 和 u-boot.bin(BL2)两文件.

根据图 2 所示, 在 Ubuntu 中使用 dd 命令烧写 tiny210-spl.bin 和 u-boot.bin 至 SD 卡. 命令如下:

Dd bs=512 iflag=dsync oflag=dsync if=spl/tiny210-spl.bin of=/dev/sdb seek=1

Dd bs=512 iflag=dsync oflag=dsync if=u-boot.bin of=/dev/sdb seek=49

插入 SD 卡至开发板卡槽后, 上电后, 通过 COM0 输出到 SecureCRT 的 U-boot 启动如图 5 所示.

```
U-Boot 2013.01-rc2 (Mar 30 2014 - 23:43:23) for TINY210V2

CPU:    S5PV210@1000MHz

Board:   TINY210V2
DRAM:  512 MiB
WARNING: Caches not enabled

PWM Moudle Initialized.
GPDOCON  : 1111, GPDODAT  : e
NAND:  2048 MiB
MMC:   SAMSUNG SD/MMC: 0, SAMSUNG SD/MMC: 1
In:    serial
Out:   serial
Err:   serial
Net:   dm9000
Hit any key to stop autoboot:  0
[Uboot-TINY210]#
```

图 5　U-boot 启动

通过 tftp 的方式加载宿主机上的内核以及 NFS 的方式挂载根文件系统. 配置好 tftp 和 NFS, 然后设置 U-boot 启动参数: tftp 加载及启动内核命令参数 setenv bootcmd “tftp 21000000 uImage; bootm 21000000”, 但需注意的是设置机器码环境变量 setenv machid=0xd8a, 否则无法正常加载内核, 然后设置 NFS 挂载根文件系统参数 setenv bootargs root=/dev/nfs rw nfsroot=192.168.1.220:/opt/roo-tfs/sysroot ip= 192.168.1.230 :::::eth0:off console=ttySAC0,115200. 设置完成命令后, U-boot 启动自动加载内核和根文件系统. 引导内核加载过程如图 6 所示. 最后启动 NFS 挂载上了 yaffs2 根文件系统, 部分过程如图 7 所示.

```
U-Boot 2013.01-rc2 (Mar 30 2014 - 23:43:23) for TINY210V2

CPU:    S5PV210@1000MHz

Board:   TINY210V2
DRAM:  512 MiB
WARNING: Caches not enabled

PWM Moudle Initialized.
GPDOCON  : 1111, GPDODAT  : e
NAND:  2048 MiB
MMC:   SAMSUNG SD/MMC: 0, SAMSUNG SD/MMC: 1
In:    serial
Out:   serial
Err:   serial
Net:   dm9000
Hit any key to stop autoboot:  0
dm9000 i/o: 0x88001000, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 00:40:5c:26:0a:5b
operating at 100M full duplex mode
Using dm9000 device
TFTP from server 192.168.1.220; our IP address is 192.168.1.230
Filename 'uImage'.
Load address: 0x21000000
Loading: T ################################################################
         ################################################################
         ################################################################
         ################################################################
         ####################
done
Bytes transferred = 4112316 (3ebfbc hex)
## Booting kernel from Legacy Image at 21000000 ...
   Image Name:   Linux-3.0.8
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    4112252 Bytes = 3.9 MiB
   Load Address: 20008000
   Entry Point:  20008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK
Using machid 0xd8a from environment

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[    0.000000] Initializing cgroup subsys cpu
```

图 6　U-boot 引导内核加载过程

```
[    3.676662] dm9000 dm9000: eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1
[    3.681482] IP-Config: Guessing netmask 255.255.255.0
[    3.681793] IP-Config: Complete:
[    3.681834]      device=eth0, addr=192.168.1.230, mask=255.255.255.0, gw=255.255.255.255,
[    3.681910]      host=192.168.1.230, domain=, nis-domain=(none),
[    3.683349]      bootserver=255.255.255.255, rootserver=192.168.1.220, rootpath=
[    5.046796] VFS: Mounted root (nfs filesystem) on device 0:12.
[    5.046960] Freeing init memory: 188K
Mounting /proc .................................Done.
Mounting /sys ..................................Done.
Mounting /dev ..................................Done.
Starting mdev ..................................[    7.490238] FAT-fs (mmcblk0p1): utf8 is not a recommended IO charset for FAT fil
esystems, filesystem will be case sensitive!
[    7.490249]
[    7.491310] FAT-fs (mmcblk0p1): Invalid FSINFO signature: 0x43503553, 0xe5801010 (sector = 1)
Done.
Mounting file system specified in /etc/fstab ....Done.
Try to bring lo interface up........Done.
Try to bring eth0 interface up......done.

Please press Enter to activate this console.
Starting telnetd daemon ........................Done.
Starting vsftpd daemon .........................Done.
```

图 7　NFS 挂载 Yaffs2 根文件系统启动过程

4　结语

Bootloader 是操作系统与硬件之间的桥梁, 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. 实验表明 U-Boot 2013.01-rc2 移植到基于 S5PV210 的 Tiny210V2 目标板上的工作已经结束, 并且可以稳定在开发板上运行, 同时还可以通过 TFTP 的方式加载内核 uImage 映像, 以及通过 NFS 的方式挂载启动 yaffs2 根文件系统, 从而完整启动 Linux. 为后续的嵌入式系统开发提供良好的软

硬件环境.

参考文献

1 武杰,黎敬涛.U_boot 在 ARM9 上的移植分析与实现.微计算机应用,2011.

2 师磊.U_Boot 在 S3C2440 上的分析与移植.计算机系统应用,2010,19(4).

3 Samsung Electronics. S5PV210 RISC Microprocessor User's Manual, Revision 1.10. Republic of Korea: Samsung, 2010.

5 U-Boot-2013.1.rc U-boot 源代码.http://ftp.denx.de/pub/u-boot/.

6 min210_linago for U-boot 源代码.https://gitorious.org/opencsbc/u-boot/source/.