基于 MPC8313E 嵌入式系统 U-Boot 的移植

陈会全，邢开宇，夏开华

（西安电子科技大学 电子工程学院，陕西 西安 710071）

摘 要：针对自行开发的基于 MPC8313E 的大容量固态存储管理系统，利用 ELDK 开发套件与 Freescale CodeWarrior 集成开发环境，给出了对 U-Boot-1.3.0 进行了开发移植的实现方法。文中主要描述了 U-Boot 源码树的结构和在 MPC8313E 上移植的方法及过程，简单阐述了 U-Boot 的结构和交叉开发环境的构建过程。

关键词：U-Boot；移植；嵌入式系统；MPC8313E

中图分类号：TP311.53 **文献标识码**：A **文章编号**：2095-1302(2011)02-0085-04

Transplantation of U-Boot Embedded System Based on MPC8313E

CHEN Hui-quan, XING Kai-yu, XIA Kai-hua

(College of Electronic Engineering, Xidian University, Xi'an 710071, China)

Abstract: The U-Boot-1.3.0 was transplanted by using ELDK and Freescale CodeWarrior integrated development environment for the MPC8313E-based large capality solid-state storage and management system. The structuse of U-Boot source tree and the method of transplantation on U-Boot are described in detail. A cross development environment and structuse of U-Boot are discussed briefly.

Keywords: U-Boot; transplantation; embedded system; MPC8313E

0 引 言

进入 21 世纪以来，以计算机技术、通信技术和软件技术为核心的信息技术取得了迅猛的发展，各种装备与设备上的嵌入式计算与系统的广泛应用，大大地推动了行业的渗透性应用。嵌入式系统由嵌入式硬件和嵌入式软件两部分组成。硬件是支撑，软件是灵魂，几乎所有的嵌入式产品都需要嵌入式软件来提供灵活多样、而且应用特制的功能。由于嵌入式系统应用广泛，嵌入式软件在整个软件产业中占据了重要地位，并受到世界各国的广泛关注；如今已成为信息产业中最为耀眼的“明星”之一。

Bootloader 是在操作系统内核运行之前运行的一段小程序。通过这段小程序可以初始化硬件设备、建立内存空间映射图，从而将系统的软硬件环境带到一个合适状态，以便为最终调用操作系统内核准备好正确的环境。U－Boot 身为 Bootloader 的一种，其功能最多，灵活性最强，能够支持 PowerPC，ARM，MPIS，X86 体系架构的上百种开发板，并可引导加载 Linux，VXWorks，QNX 等多种操作系统，有丰富的开发调试文档资料与强大的网络技术支持。本文针对自行开发的基于 MPC8313E 的大容量固态存储管理系统，利用 ELDK 开发套件与 Freescale CodeWarrior 集成开发环境，给出了对 U-Boot-1.3.0 进行开发移植具体方法。

1 硬件平台简介

基于 MPC8313E 的大容量固态存储管理系统的组成框图如图 1 所示。

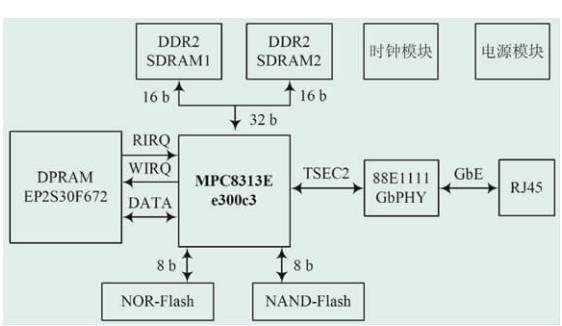

图 1 大容量固态存储管理系统原理框图

该系统采用飞思卡尔 MPC8313E 芯片能以极低的价位，来简化大量高速外围设备的部署。除了 GigE 与 USB2.0 w/ phy 集成之外，MPC8313E 还提供有 32 位的双倍数据速率(DDR1/DDR2)存储器控制器、16 位局部总线和 4 个直接存储器访问(DMA)通道[1]。

本系统可以使 MPC8313E 从 FPGA 中的双口

RAM 中高速读取数据，然后通过千兆网将其上传至上位机并接受上位机的相关命令，最后将解析命令并下发给 FPGA。

2 U-Boot 简介

U-Boot 全称为 Universal Boot Loader，是遵循 GPL(General Public License)条款的开放源码项目。起初，DENX 软件工程中心的 Wolfgang Denk 基于 8xxrom 的源码创建了 PPCBOOT 工程，并且不断的添加处理器支持。后来，Sysgo Gmbh 把 ppcboot 移植到 ARM 平台上，并创建了 ARMBOOT 工程。然后以 PPCBOOT 工程和 ARMBOOT 工程为基础，创建了 U－Boot 工程。目前，U-Boot 仍然由 DENX 的 Wolfgang Denk 等人维护，从 DENX 的官方 ftp 中可知，U－Boot 的最新稳定版本为 U-Boot-2010.12。

2.1 U-Boot 相关源码结构

系统采用 U－Boot 的版本为 1.3.0，它支持 Freescale 的 MPC8313ERDB 开发板，其中与移植相关的主要目录如表 1 所列。

表 1 U-Boot 源码结构

目录	特性	解释说明
board	平台相关	目标板相关代码。相关目录：freescale\mpc8313erdb
cpu	平台相关	目标处理器相关代码。相关目录：mpc83xx
lib_ppc	平台相关	PowerPC 处理器通用库文件。
lib_generic	平台无关	通用库函数文件。
include	平台无关	头文件和目标板配置文件。相关目录：configs
common	平台无关	通用多功能函数的实现，U－Boot 命令的实现代码。
drivers	平台无关	设备驱动程序代码。
net	平台无关	网络相关代码。

2.2 U-Boot 启动流程

硬件复位中断向量位于 NOR-Flash 的 0x0100 地址(即 0xFE00 0100)。

当 MPC8313E 上电复位后，CPU 将根据 RST__CFG_SRC[0:3]的值从数据总线上读取硬件配置字或者采用默认硬件配置字，根据配置字来设置相应的时钟频率、器件等。之后转向硬件复位向量，其具体步骤如下：

(1) 从 U-boot 中的 cpu/mpc83xx/start.s 的__start 开始执行，主要是初始化 CPU 的一些内部寄存器的状态；

(2) 之后随着 bl cpu_init_f 跳转到 cpu/mpc83xx/cpu_init.c 中的 cpu_init_f()函数开始对 CPU 底层进行初始化，并在退出时将地址返回到 r3 寄存器中(mr r3,r21)；

(3) 在 CPU 底层初始化完成程序返回 start.s 后，紧跟着是 bl board_init_f 跳转到 lib_ppc/board.c 中的 board_init_f()函数，对目标板的第一次初始化，以完成对 RAM 的初始化，并分配内存空间；

(4) 程序再次返回 start.s 中进行必要的设置，直到执行到 relocate_code，再代码搬运到 DDR2 SDRAM 中，并调整全局偏移量表(GOT)，对存储空间进行重定位；

(5) 当代码搬运完毕后，程序将在 DDR2 SDRAM 中运行，此时调用 lib_ppc/board.c 中的 board_init_r()可对目标板进行第二次初始化，以完成一些数据结构、高端模块和系统设备的初始化；

(6) board_init_r 初始化完毕，调用 commom/main.c 中的 main_loop()函数并等待用户输入命令，同时通过调用 run_command()函数对命令进行解析已完成相应的操作；

(7) 至此，U-boot 启动完成。

3 U-boot 的移植

对于 U-Boot 的移植可采用主机－目标板模式，开发主机是 X86 构架下的 Windows XP SP3 操作系统，并通过虚拟机 VMware 安装 Linux 操作系统(Red Hat 9.0)，目标板是 PowerPC 构架下的 Linux 操作系统。主机和目标板之间可通过串口与网络进行通信，其连接方式如图 2 所示。

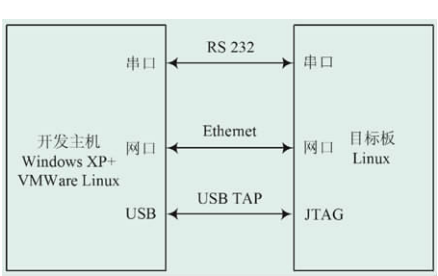

图 2 开发主机和目标板之间的通信接口

3.1 交叉开发环境的构建

交叉编译是指在一种计算机环境中运行的编译程序能编译出在另外一种环境下运行的代码。简单地说，就是在一个平台上生成另一个平台上的可执行代码。这里需要注意的是，所谓平台，实际上包含两个概念：体系结构和操作系统。同一个体系结构可以

运行不同的操作系统；同样，同一个操作系统也可以在不同的体系结构上运行。

由于开发主机使用的是 X86 构架的 Linux，目标板是 PowePC 构架，我们需要首先搭建交叉的编译环境。ELDK(Embedded Linux Development Kit)是德国 denx 提供的可供 PowerPC 嵌入式 Linux 移植的完整开发环境编译套件，在开发主机上构建交叉编译环境的主要过程[2]如下：

(1) ./install －d /usr/ppc/eldk ppc_6xx；

(2) 编辑自己的帐户目录，并在. bashrc (例如：/root/) 中加入下面内容：

```
CROSS_COMPILE=ppc_6xx－
PATH= $ PATH：/usr/ppc/eldk/usr/bin:/usr/ppc/eldk/bin
export CROSS_COMPILE PATH
```

(3) 保存。然后执行 source . bashrc

(4) 验证安装，执行下列命令：

powerpc－linux－gcc －v

同时输出 gcc 和 glibc 的版本号，例如：gcc version 4. 0. 0 (DENX ELDK 4. 1 4. 0. 0)，至此表明 ELDK 已安装成功，交叉开发环境构建完成。

3.2 U-Boot 的移植

在修改 U-Boot 代码之前，首先要做好系统地址空间的分配，而我们的目标板是由 MPC8313ERDB 开发板裁剪后并添加与 FPGA 中双口 RAM 接口产生的，所以只需要去掉 MPC8313ERDB 开发板中不用的部分并添加 DPRAM 的驱动即可，因此，设计时主要修改 U－Boot－1. 3. 0 中与 MPC8313ERDB 开发板有关的代码就可以了，这样既可提高移植后 U-Boot 的稳定性，又可缩短开发时间。目标板的系统地址空间的分配如表 2 所列[3]。

表 2 系统地址空间分配

Definition	Start	End	Size(Byte)
DDR2 SDRAM	0x0000_0000	0x07ff_ffff	128M
IMMR	0xe000_0000	0xe00f_ffff	1M
NAND flash	0xe280_0000	0xe280_7fff	32M
DPRAM	0xf000_0000	0xf001_ffff	128K
NOR flash	0xfe00_0000	0xfe7f_ffff	8M

(1) 修改顶层 Makefile[4] 文件为目标板添加新的规则：

```
MPC8313EMYB_33_config：unconfig
@mkdir －p $ (obj)include
@echo "＞"$ (obj)include/config. h ；\
echo －n ".. 33M ... "；\
echo #define CFG_33MHZ "＞＞ $ (obj)include/config. h ；
@ $ ( MKCONFIG ) － a MPC8313EMYB ppc mpc83xx mpc8313emyb
```

(2) 在 board 目录中创建 mpc8313emyb 目录，将 board/freescale/mpc8313erdb 目录中的全部文件复制到该目录中并将 mpc8313erdb. c 文件重命名为 mpc8313emyb. c。修改 mpc8313emyb. c 文件，去掉 PCI 相关代码(//mpc83xx_pci_init(1，reg，warm-boot)；)；

(3) 将 include/configs 目录中 MPC8313ERDB. h 的内容复制到 MPC8313EMYB. h 中，并作如下修改：

```
//#define CONFIG_PCI
//#define CONFIG_83XX_GENERIC_PCI
//#define CONFIG_CMD_PCI
//#define CONFIG_CMD_I2C
#define CFG_SICRL(SICRL_USBDR | 0x003C0000)
                                //使能 GPIO30 和 GPIO31
#define CFG_DPRAM_BASE0xF0000000
#define CONFIG_DPRAM_ENET
                                /*DPRAM ethernet support*/
#define CFG_BR2_PRELIM0xf0001001
                                /*DPRAM Base address*/
#define CFG_OR2_PRELIM0xfffe0033
                                /*DPRAM，128K bytes*/
#define CFG_LBLAWBAR2_PRELIMCFG_DPRAM_BASE
#define CFG_LBLAWAR2_PRELIM0x80000010
#define CONFIG_IPADDR192. 168. 0. 10
#define CONFIG_SERVERIP192. 168. 0. 155
#define CONFIG_GATEWAYIP192. 168. 0. 1
#define CONFIG_NETMASK255. 255. 255. 0
```

(4) 编译生成 U-Boot 镜像：

```
make MPC8313EMYB_config
make
```

编译完成后生成 U-boot. bin 文件；

(5) 烧写镜像到目标板上

利用 ftp 将 VMware 中 Linux 中的 u－boot. bin 文件下载到 Windows 下，用 Freescale CodeWarrior 中内嵌的 Flash Programer 将 U－Boot 镜像烧写到目标板的 NOR－Flash 中。

4 U-Boot 的调试

利用 Freescale 的 CodeWarrior 软件和 USB TAP 可以很方便地在目标板上对 U-Boot 经行调试。调试 U-Boot 首先要在其镜像文件中添加一些调试信息，在编译 U-Boot 镜像之前需要对代码树作如下修改[5]：

（1）修改 U-Boot 源码根目录下的 config. mk 文件；

```
DBGFLAGS = -g2 -gdwarf-2
AFLAGS_DEBUG = -Wa,-gdwarf2
OPTFLAGS = -O1
```

（2）在 lib_ppc/board. c 中找到 debug（"how running in ram… "）;语句并将 debug 改为 printf；

（3）编译 U-Boot，得到包含调试信息的 ELF 格式 U-Boot 文件。

利用第 3 步得到的 ELF 格式的 U-Boot 文件建立 CodeWarrior 工程。

U-Boot 的调试分为三个阶段：MMU 开启前的 NOR－Flash 运行阶段、MMU 开启后的 NOR－Flash 运行阶段和在 RAM 中的运行阶段。MPC8313E 上电复位配置后（读取完硬件复位配置字），程序指针（PC）将跳转到 BR0＋0x100 映射的地址去取第一条指令，这里就是 NOR－Flash 中的 0x100 地址（_Start:），之后会在 NOR－Flash 中运行（NOR－Flash 具有片内执行（eXecute In Place：芯片内执行）的特性）最初的初始化程序（主要是初始化 RAM），直到程序被搬运到 RAM 中。程序在 NOR-Flash 中运行阶段还可细分为两个阶段：MMU Enable 前和 MUU Enable 后。

调试完 U-Boot 后，就可以将内核镜像、扁平设备树文件和文件系统镜像分别加载到目标板上，至此，一套基于 MPC8313E 微处理器的 Linux 嵌入式系统就移植完成了。

5 结 语

U-Boot 作为一个通用的 Bootloader 软件，可以稳定的应用到多种构架的微处理器嵌入式平台上，在作者开发的基于 MPC8313E 微处理器的大规模固态存储管理系统中得到了很好的应用，已经成功的将 linux－2. 6. 23 内核和 Ramdisk 格式的文件系统移植到目标板上，并测试了 DPRAM 驱动，本套系统现已成功的在实际应用中使用。

参 考 文 献

[1] MPC8313E PowerQUICC™Ⅱ Pro integrated processor family reference manual[R]. freescale semiconductor, 2008.

[2] The DENX U-Boot and linux guide (DULG) for canyonlands. DENX Software Engineering, 2007.

[3] MPC8313E-RDB BSP user's manual. Freescale Semiconductor, 2008.

[4] 徐海兵. GNU make 中文手册[R]. ver-3. 8. 2004.

[5] CodeWarrior™ development studio for power architecture™ processors professional/linux TM application editions targeting manual. Freescale Semiconductor, 2008.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计

VxWorks：

1. 基于 VxWorks 的多任务程序设计

2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）

17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发

ARM:

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植

Hardware:

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计