PowerPC 平台引导加载程序的移植

张 磊

（第七一五研究所，杭州，310012）

摘要 引导加载程序（Boot Loader）是嵌入式系统软件开发的第一个环节，它把嵌入式操作系统和硬件平台衔接在一起，对于嵌入式系统的后续开发起到十分重要的作用。U-Boot 是目前比较流行且功能强大的 Boot Loader，支持种类繁多的体系结构，尤其对 PowerPC 提供了完善的支持。MPC8548E 是基于 E500v2 内核，并提供多种接口控制器的处理器。详细介绍了在基于 MPC8548 CPU 的 SBC8548E 开发板上 Boot Loader 的移植过程。

关键词 引导加载；程序设计；程序移植

引导加载程序（Boot Loader）是用来启动设备以及执行内核加载的系统软件组件，配置 Boot Loader 是所有嵌入式操作系统的一项必要工作。Linux 内核有许多 Boot Loader 可用，不同架构之间的 Boot Loader 的质和量也有很大的差异。Das U-Boot（一般称为 U-Boot）被认为是功能最多、最具有弹性以及开发最积极的开放源码 Boot Loader。而基于 E500 内核的 PowerPC 芯片是 Freescale 公司未来高端通信领域的发展方向。因此研究分析基于 E500 内核的 U-Boot 对于深入理解 PowerPC 架构以及开发后续嵌入式操作系统 Linux 具有积极的意义。

1 U-Boot 移植过程分析

MPC8548E 是基于 E500v2 内核的处理器，支持 36-bit 实地址模式（物理寻址能力达到 64 GB，虚拟寻址能力达到 1TB）和双精度浮点运算[1]，支持 DDR2 SDRAM、PCI-X、PCI-E 和 RapidIO 高速串行总线。主要适用在收发基站（BTS）、媒体网关（MGW）、无线网络控制（RNC）、刀片服务器等网络通信高端领域，同时在工业控制领域也有着广泛的应用。但是对于应用程序开发人员来说，要想充分发挥其性能就需要深入理解 E500 内核和外围器件接口的全部寄存器配置和外围总线使用方法，开发难度较大。因此需要在此平台上运行一种嵌入式操作系统（如 Linux），以封装全部硬件细节，提供给用户标准的 API 接口，方便用户程序开发调试工作。但是由于操作系统本身没有自举能力，我们需要首先运行引导加载程序（如 U-Boot），进行硬件设备初始化工作，建立内存空间的映射图，为加载操作系统内核准备好正确的环境。

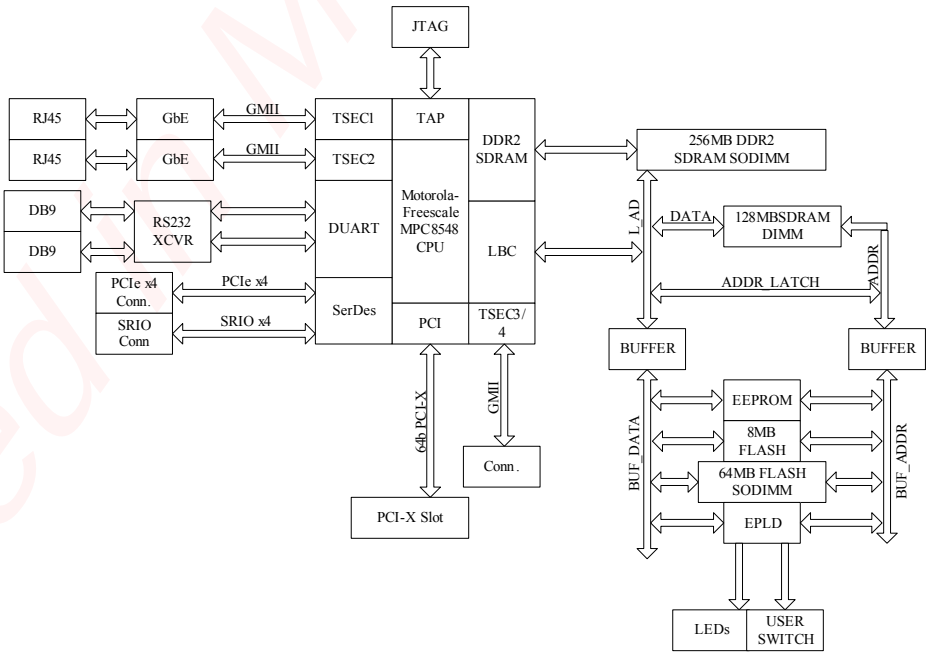

图 1 SBC8548E 开发板原理框图

Freescale 公司 PowerPC 处理器一般使用 U-Boot 作为引导加载程序。U-Boot 的前身是德国 DENX 公司的 PPCBOOT 项目，最初仅支持 PowerPC 系列处理器，目前已经成为社区维护的开源项目，支持 PowerPC、x86、MIPS、ARM、Blackfin 等诸多常用系列的处理器，并提供对 Linux、VxWorks、OpenBSD、QNX、pSOS、LynxOS 等目标操作系统的支持。

我们要移植的目标板是 Windriver 公司的 SBC8548E 开发板[2]，开发板原理框图如图 1 所示。这块开发板的特征如下（仅列出与移植相关部分）：MPC8548E 处理器、256MB DDR2 SDRAM、128 MB SDRAM、8 MB on-board flash、Gigabit Ethernet、RS-232 等。下文将详细介绍 U-Boot 在 SBC8548E 开发板的移植过程。

1.1 移植内存管理单元

内存管理单元 MMU 是处理器的重要组成部分，主要功能是将程序使用的虚拟地址转换为处理器能够直接访问的物理地址。E500 内核 MMU 中共使用了两个查找表 TLB0（Translation lookaside buffer）和 TLB1 来实现虚拟地址与物理地址的转换。其中 TLB0 一共有 256 个 Entry，每个 Entry 映射 4 kB，用来进行页式映射；TLB1 一共有 16 个 Entry，段大小可以调整，用来实现段式映射。

U-Boot 启动分为阶段 1 和阶段 2 两大部分，依赖于 CPU 体系结构的代码（如 CPU 初始化等）通常都放在阶段 1 中且通常用汇编语言实现，而阶段 2 则通常用 C 语言来实现，这样可以实现复杂的功能。为了构建一个 C 语言环境，我们就需要分配一个栈，但是系统在上电后，由于内存控制器未初始化，因此主存储器无法使用，也就没有数据段和 BSS 段，所以我们首先利用在 MPC8548E 中的二级 cache 分配了一个 16 kB 的栈。并用 TLB0 的 4 个 Entry 进行页式映射。然后使用 TLB1 对开发板上的硬件资源进行段式映射。要注意在进行段式映射时，虚拟地址不能重叠。如果某 PCI Memory 空间为 512 MB，由于寄存器 TSIZE 为……64 MB、256 MB、1 GB、2 GB，因此要用 2 个 TSIZE 为 256 MB 的段拼接构成 PCI Memory 段空间，而不能配置成为 1 GB 的 PCI Memory 段空间。由于 SBC8548E 开发板 Memory Map 比较简单，在系统软件设计中仅需要使用 TLB1 完成虚实地址空间的转换，不需要使用 TLB0 进行页式虚实地址转换。

1.2 移植 DDR2 控制器

SDRAM 控制器的配置参数一般是通过读取 SDRAM 模块上的 SPD 获得。SPD 是 SDRAM 模块上一个单独的 EEPROM，它存储了 SDRAM 模块的存储密度、时序和性能参数；当系统上电后，Boot Loader 通过预设地址检测 SPD，然后读取配置参数对 SDRAM 控制器进行配置；但是在工业环境中，DIMM 封装的 SDRAM 模块可靠性不高，我们一般通过板载焊接方式安装 SDRAM。因此我们就需要深入了解 SDRAM 的时序，尤其要注意 DDR2 SDRAM 相对于传统 SDRAM 的新特性，如 Additive Latency、Differential Signals、Two Output Driver Levels、On-Die Termination。一般来说我们需要特别关注 CSx_BNDS、CSx_CONFIG、TIMING_CFG_1、TIMING_CFG_2、DDR_SDRAM_CFG、DDR_SDRAM_MODE、DDR_SDRAM_INTERVAL 等寄存器的配置。

1.3 移植其它外围接口

在完成了上述两项核心工作后，按照系统启动顺序，我们首先要配置 Flash 芯片。在系统上电后，CPU 首先从 Flash 芯片中拷贝代码到 RAM 中，然后在 RAM 中执行代码。由于目前常用的 Flash 芯片都是 CFI（Common Flash Interface）兼容，芯片都会在固定地址存储芯片特定信息，因此 U-Boot 通过读取特定信息，获得芯片类型，然后通过查表得知其详细配置参数，再对 Flash 接口寄存器进行配置，Flash 即可正常工作。

下一个要配置的为串口寄存器。由于目前常用的串口芯片的编程模型符合 PC16550 UART，MPC8548E 上的 DUART 模块也不例外，这样就大大简化了串口配置工作，仅需要确定其寄存器基地址和接口数据宽度，通用串口驱动程序就可以正常工作。

最后进行网络配置。MPC8548E 共有 4 个 eTSEC（Enhanced Three-Speed Ethernet Controller）。考虑到网络冗余，我们在 U-Boot 中配置 2 个 eTSEC。由于 MPC8548E 没有集成 PHY，需要外接 PHY，因此对网络的配置工作主要在于对 PHY 地址的配置。当正确配置 PHY 地址后，网络控制器也就可以收发数据了。

完成以上这些工作，MPC8548E 的 Boot Loader 已经可以加载内核，同时可以进行上电自检和运行简单应用程序等工作，软件可以对硬件进行大规模、有强度的测试。

2 U-Boot 引导 Linux 初始化

Linux 系统的引导一般都要分为三大步骤，如图 2 所示。

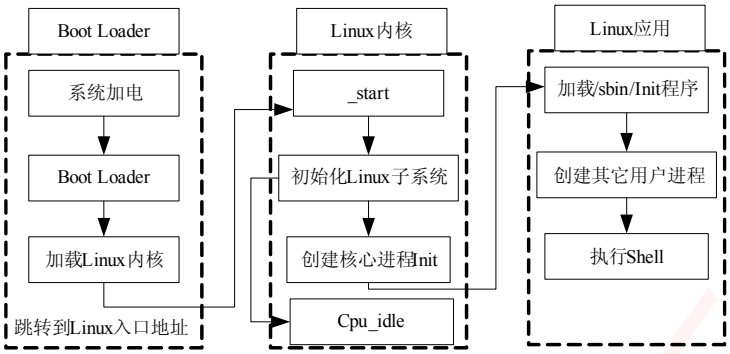

图 2 Linux 系统的引导步骤

对嵌入式系统底层开发人员而言，关心的是 Boot Loader 如何引导 Linux 内核初始化，即 Boot Loader 的程序如何与 Linux 内核之间进行信息传递。最新的 U-Boot 和 Linux 源代码使用 OF 树形结构对一个处理器系统进行描述，该结构由 Open Firmware Working Group 提出，并成为 IEEE1275-1994 标准[3]。在 E500 内核中，由于 MMU 不能关闭，程序不能直接访问物理地址，因此在 MPC8548E 中，使用通用寄存器 r4 保存 Linux 内核所在的有效地址，使用通用寄存器 r3 指向 OF 结构的有效地址。在内核获得存储器控制之前，内核的一段代码在 U-Boot 的上下文环境中执行，扫描设备树获得设备信息并传递到内核中。例如 MPC8548 的网络接口子设备树结构如下：

```
ethernet@24000 {
    #address-cells = <1>;
    #size-cells = <0>;
    device_type = "network";
    model = "eTSEC";
    compatible = "gianfar";
    reg = <24000 1000>;
    local-mac-address = [ 00 E0 0C
00 73 00 ];
    interrupts = <d 2 e 2 12 2>;
    interrupt-parent = <40000>;
    phy-handle = <2452000>;
};
```

结构中属性的详细介绍参见./Documemtion/powerpc/boot-without-of.txt。内核通过读取该设备树，可以获得网络接口模块的硬件信息以及驱动程序的特定参数，使驱动程序能够正常工作。在内核获得处理器系统全部的硬件信息后，即可正确加载和配置相应的设备驱动程序，嵌入式操作系统的启动部分就可以正常工作了。

3 结束语

本文是作者在实际开发过程中根据相关资料进行摸索，并在成功移植了 U-Boot 和 Linux 的基础上总结出来的。目前已在 SBC8548E 开发板上成功运行，由于 PowerPC E500 系列处理器种类众多，且多核产品即将量产，必将在控制和信号处理领域拥有广泛的应用前景。对于基于 E500 内核不同的处理器和系统平台，其开发的基本方法和步骤是相同的，希望本文能对相关嵌入式系统的设计人员有所帮助。

参考文献：

[1]Freescale Inc. MPC8548E PowerQUICC III Integrated Processor Family Reference Manual,2007.

[2] Windriver Inc. SBC8548E Board User Reference Manual, 2007.

[3] David Gibson,Benjamin Herrenschmidt. Device trees everywhere[EB/OL]. 2006-2-13. [2008-1-9]].http:// ozlabs. org /~dgibson /dtc /dtc.git

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
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较

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
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计

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
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现

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
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现

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
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计

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
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计

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
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究

Programming:

1. 计算机软件基础数据结构 - 算法