第 36 卷 第 6 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.6 Computer Engineering March 2010
·开发研究与设计技术· 文章编号：1000—3428(2010)06—0267—03 文献标识码：A 中图分类号：TP274

嵌入式 Bootloader 机制的分析与移植

王亚刚
(西安邮电学院计算机科学与技术系，西安 710121)

摘 要：由于嵌入式系统的专用性和多样性，嵌入式启动引导程序(Bootloader)的移植工作繁琐复杂。为了减少 Bootloader 移植工作的盲目性，加速产品进入市场的时间，以 U-Boot-1.1.1 为例，对嵌入式 Bootloader 进行分析，结合 HHPPC852T 嵌入式开发板的具体情况，提出在嵌入式系统上移植 Bootloader 的一般方法，该方法简单可行。

关键词：嵌入式系统；启动引导程序；移植

Analysis and Transplant of Embedded Bootloader Mechanism

WANG Ya-gang
(Department of Computer Science and Technology, Xi'an Institute of Post and Telecommunications, Xi'an 710121)

【Abstract】 Due to the specialty and diversity of the embedded systems, much tedious transplant work must be done when embedded Linux is deployed on the target embedded system. In order to overcome the blindness of transplant work and speed up the entry markets time of the embedded products, a brief introduction to the embedded system Bootloader(U-Boot as an example) is given, with emphasis on its basic structure and mechanism. Based on the implementation of transplanting U-Boot to HHPPC852T target, a typical method for transplanting Bootloader to a new target is presented, which is simple and practicable.

【Key words】 embedded system; Bootloader; transplant

1 概述

Bootloader 是嵌入式系统在引导操作系统内核或用户应用程序之前运行的一段程序，其主要功能是完成处理器和周边电路正常运行所需要的初始化工作，建立内存空间的映射(包括设置系统堆栈和系统启动参数区等)，从而将系统的软硬件环境带到一个合适的状态，并从 ROM, Flash 等非易失存储器上，或者从网络上加载操作系统映像文件(或者用户的应用程序映像)，以便最终引导操作系统或执行用户应用程序。

然而对于嵌入式系统来说，由于其固有的专用性和多样性，通常都缺少一个类似于 PC 机上 BIOS 的通用底层软件支持，因此，嵌入式系统上的 Bootloader 的功能要求更加多样化，不仅包括从存储介质中读取操作系统并加以执行，还必须负责最初的硬件初始化和一些硬件的检测功能(即 PC 机上 BIOS 所完成的部分功能)。

通常，Bootloader 是高度依赖于 CPU 体系结构等硬件环境而设计和实现的。另外，除了依赖于 CPU 的体系结构外，Bootloader 的实现也依赖于具体的嵌入式板卡设备的配置，比如板卡的硬件地址分配情况、RAM 芯片的类型及其大小、Flash 芯片的类型以及与其片选(Chip Select, CS)的设置等。

2 Bootloader 的一般结构

为了满足 Bootloader 的专有性和灵活性需要，大多数 Bootloader 的代码都由 2 个阶段完成，即 stage1 和 stage2[1]。其中，依赖于 CPU 体系结构的代码，例如 CPU 及存储管理部件初始化代码等，通常都放在 stage1 中，并用汇编语言来实现，以达到短小精悍的目的；stage2 则通常用 C 语言等高级语言来实现，这样可以实现比较复杂的功能，并且使程序具有更好的可读性和可移植性。

Bootloader 的 stage1 通常包括硬件设备初始化，例如配置处理器中的关键寄存器和一些必要的硬件参数，为加载 Bootloader 的 stage2 代码准备 RAM 空间，拷贝 Bootloader 的 stage2 的代码到 RAM 空间中，设置好堆栈并跳转到 stage2 的 C 程序入口点。

Bootloader 的 stage2 通常包括以下步骤：初始化本阶段要使用到的硬件设备，检测系统内存映射(memory mapping)，将操作系统映像文件从 Flash 等非易失性存储器或者网络文件系统加载到 RAM 空间中，为操作系统设置启动参数，跳转到操作系统入口执行。

3 U-Boot 分析

3.1 U-Boot 简介

U-Boot(Universal Bootloader)是遵循 GNU GPL 条款而开发的开放源代码项目，是从 8xxrom, PPCBoot 等 Bootloader 逐步发展演化而来，其源代码目录、编译形式与 Linux 内核非常相似[2]。U-Boot 中 Universal 具有 2 层含义：(1)U-Boot 支持 Linux, NetBSD, VxWorks, QNX, RTEMS, ARTOS, LynxOS 等嵌入式操作系统；(2)U-Boot 支持 PowerPC, MIPS, x86, ARM, XScale 等诸多常见的处理器系列。这 2 个特点正是 U-Boot 项目的开发目标，即支持尽可能多的嵌入式处理器和嵌入式操作系统。U-Boot 源码开放，支持多种嵌入式操作系统，支持多个处理器系列，具有较高的可靠性和稳定性，同时可以进行灵活的功能设置，便于进行调试和产品发布等。另外 U-Boot 源码中有丰富的设备驱动程序源码，如串口、以太网、SDRAM、Flash、LCD、NVRAM、EEPROM、RTC、

基金项目：陕西省科技创新专项基金资助重大项目(2008ZKC02-11)
作者简介：王亚刚(1972－)，男，讲师、博士研究生，主研方向：嵌入式系统，SoC 设计
收稿日期：2009-11-10 **E-mail：**wangyg@xiyou.edu.cn

键盘等，同时也提供丰富的开发调试文档与强大的网络技术支持。正是基于以上特点，目前 U-Boot 已被广泛地应用到嵌入式系统产品开发之中。

3.2 U-Boot 源码结构

U-Boot 的源码结构与 Linux 内核源码的组织方式比较类似，主要包括以下主要目录：(1)board：目标板相关文件，主要包含 SDRAM, Flash 驱动等。(2)common：独立于处理器体系结构的通用代码，如内存大小探测等，另外 U-Boot Monitor 中的交互命令一般也在这里定义。(3)cpu：处理器体系结构相关的实现部分，主要包括 CPU 的初始化、中断初始化及与处理器相关的硬件驱动等，如 mpc8xx 子目录下包括串口、SCC、FEC、I2C 及 LCD 等驱动程序及等文件。(4)driver：通用设备驱动，如 CFI, Flash 驱动。(5)include：U-Boot 源代码头文件，其中的 configs 子目录下包含了许多 U-Boot 支持的标准板的硬件配置选项，这些头文件在移植过程中经常需要修改。(6)lib_xxx：处理器体系相关的通用程序的实现部分，例如 lib_ppc 及 lib_arm 目录分别包含了与 PowerPC, ARM 体系结构相关的通用程序的实现。(7)net：与网络功能相关的文件目录，如 bootp, nfs, tftp。(8)tools：用于创建 U-Boot S-RECORD 和 BIN 镜像文件的工具。

3.3 U-Boot 工作过程分析

以 U-Boot-1.1.1 为对象，分析 U-Boot 引导 HHPPC852T 标准板的过程。根据 U-Boot 运行的环境，U-Boot 的启动过程大致可以分成 2 个部分：在 Flash 中运行的阶段与在 RAM 中运行的阶段。cpu/mpc8xx/start.S 是 U-Boot 启动的起点，用 PowerPC 汇编语言编写，入口为_start。

第 1 阶段：U-Boot 在 Flash 中运行。这一阶段主要完成 CPU 寄存器的设置及板卡的部分硬件初始化，包括获取 CPU 和总线时钟、波特率发生器的初始化、串口及终端的初始化、CPU 的检测、板卡的检测、DRAM 的检测等，最后 U-Boot 调用 relocate_code 把自己的运行代码从 Flash 搬运到 RAM 中，并跳转到 in_ram 处继续执行 U-Boot 第 2 阶段的代码。

第 2 阶段：U-Boot 在 RAM 中运行。这一阶段 U-Boot 运行在 RAM 中，主要进行进一步的初始化和设置，包括重新定位交互命令的地址、trap 的初始化、Flash 初始化；初始化 malloc 内存区域；重定位环境函数指针；设置 Ethernet 的 MAC 地址；设备初始化；应用程序跳转表；初始化终端；初始化中断；设置时钟；Ethernet 初始化等。最后 U-Boot 调用函数 main_loop()自动加载操作系统或者进入用户交互环境。

U-Boot-1.1.1 中主要函数之间的调用关系如图 1 所示。

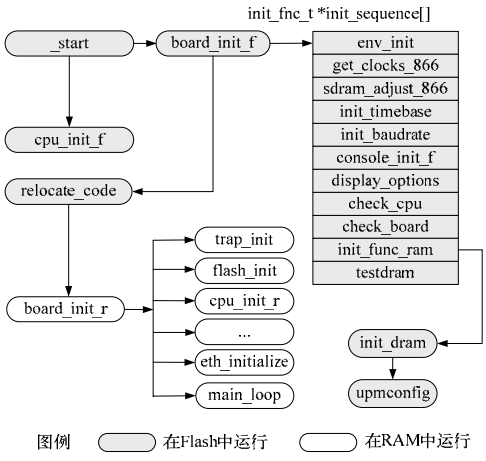

图 1 U-Boot 主要函数调用关系

4 U-Boot 的移植

尽管 U-Boot 本身支持众多的 CPU 体系结构及操作系统，但由于嵌入式系统硬件设计千差万别，因此要在特定的目标板上正确地运行 U-Boot 还需要做适当的移植修改工作。一般来说，系统差异可能存在于以下几个方面：用户定制的硬件设计与 U-Boot 所支持的硬件不相同或者有差异；用户使用的 Flash 类型还未被 U-Boot 所支持；用户需要增加新的硬件调试功能；用户需要用 U-Boot 直接引导自定义的应用程序。

4.1 移植方法与移植环境

当前，对于 U-Boot 的移植方法，大致分为 2 种[3]。一种是用 BDI2000 等在线调试器(On-Chip Debugger)创建目标板初始运行环境，将 U-Boot 映像文件 u-boot.bin(RAM 版本)下载到目标板 RAM 中的指定位置，然后使用调试器进行跟踪调试。其优点是无需每次都将 U-Boot 镜像文件烧写到 Flash 中，可以减少 Flash 的擦写次数，延长 Flash 芯片的使用寿命，但是由于被调试的 U-Boot 映像文件是从 RAM 中开始运行的，和 U-Boot 从 Flash 中启动的实际过程不一致，因此调试的结果和实际运行的情况有一定差别。另外一种被广泛采用的方法是先用 BDI2000 等调试器将 U-Boot 映像文件烧写到 Flash 中去，然后从 Flash 中开始运行，并使用 BDI2000 或 BDM 调试器进行跟踪调试。这种方法所用调试器的配置文件较为简单，调试过程与 U-Boot 移植后运行过程相吻合，即 U-Boot 先从 Flash 中运行，然后把自身重新搬动至 RAM 中的新位置，并从那里正式投入运行。此方法唯一的缺点就是需要不断地擦写 Flash 芯片，影响 Flash 的使用寿命。但考虑到 Flash 常规擦写次数基本为 10 万次左右，作为 U-Boot 移植，不会使用太多的次数，所以可以忽略这个因素。

在进行 U-Boot-1.1.1 移植的过程中，使用的移植环境如图 2 所示。其中，开发主机上安装 Redhat Linux 9.0 操作系统及移植 U-Boot 所需要的交叉开发环境 ELDK3.0。为了给目标板提供 U-Boot 映像的网络下载，可以在开发主机上开启 TFTP 服务。同时开发主机上运行 NFS 服务器及 DHCP 服务器程序，可为开发板的运行系统提供 NFS 文件系统及 DHCP 服务。串行终端作为移植及调试过程中目标板系统信息输出的主要途径，可以使用 Windows 中的 HyperTerm 或者 Linux 下的 minicom 等终端程序。连接开发主机和目标板的 BDM 调试接口是系统调试的主要途径。

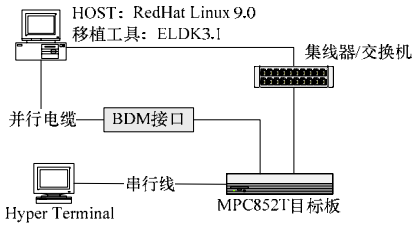

图 2 移植环境

4.2 移植的一般过程

对于 U-Boot 的移植，一般包括以下几个步骤：获得最新发布的 U-Boot 源码版本；建立 GNU 交叉开发环境；根据移植目标板的硬件配置确定 U-Boot 移植时的参考板类型；根据移植目标板的硬件配置对 U-Boot 源代码进行修改，并重新编译；在目标板与开发主机间接入硬件调试器进行调试。

其中，移植参考板的类型选择尤为重要，这是进行 U-Boot 移植首先要明确的一个关键问题。一般情况下为特定的开发板移植 U-Boot 并不需要从头开始，因为在 U-Boot 已

经支持的众多的开发板中，总会有一些标准板与用户的开发板配置比较相似，通过参考这些已有开发板的程序和配置文件，用户可以大大减少移植的工作量。因此，必须根据目标板上 CPU 的体系结构、Flash、SDRAM 等存储器的硬件配置等情况，以尽可能相一致为原则，找出一个与移植目标板相同或相近的、U-Boot 所支持的标准板的类型作为移植的参考板。

结合本文所介绍的 U-Boot 移植，移植时参考板的选择可以基于以下步骤：

(1)明确目标板硬件的配置。对于采用 Motorola 处理器的嵌入式目标板来说，几乎每种目标板都有其独特的设计，包括处理器类型、内存地址映射等。例如，处理器可以选择 MPC860, MPC866 或者 MPC852T 等；有些目标板使用 SMC1 作为串行口，有些则使用 SMC2 作为串行口，有些板卡使用 SCC3 作为以太网接口，有些则使用 SCC4 等。因此，作为移植的依据，在把 U-Boot 移植到目标板的过程中，必须了解目标板的一些基本硬件配置，主要包括 CPU 型号、频率等；Flash, SDRAM 等存储器的型号、容量、数量以及片选等；串口通道选择及波特率、以太网控制器及端口选择、FEC 控制器及端口选择等。

表 1 描述了与 U-Boot 移植密切相关的目标板 HHPPC852T 的硬件配置信息。如果对开发板的硬件配置不清楚，那么移植的工作将会非常困难，因此，需要与硬件工程师密切配合并寻求帮助。

表 1　目标板硬件配置

硬件	配置信息
CPU	型号 MPC852TZT100，晶振频率 10 MHz，CPU 频率 100 MHz，倍频数 10，IMMR 地址 0xFFF00000，串行口 SMC1，串行口波特率 115 200 Baud/s，10 M Ethernet SCC4，100 M Ethernet FEC
SDRAM	型号 Hynix HY57V641620HG T-6，容量 16 MB，起始地址 0x00000000，片选 CS2
Flash	型号 AMD AM29LV-160DB-90EC，数据宽度 16，单片容量 1 M×16 bit，单片扇区数 35，Flash 数量 2 片， Bank 1：起始地址 0x40000000 片选 CS0，Bank 2：起始地址 0x40200000 片选 CS1

(2)根据移植目标板的硬件配置，确定移植参考板的类型。本文在进行 U-Boot 移植时选取参考板的类型为 TQM866M，主要基于以下原因：

1)U-Boot 不支持 HHPPC852T 标准开发板；

2)U-Boot 支持 TQM866M 标准开发板。

Motorola 处理器 MPC866P, MPC866T, MPC859P, MPC859T, MPC859DSL, MPC853T, MPC852T 在体系结构上是一致的。另外，MPC866 及 MPC852T 等这些处理器都采用全新的、与 MPC860 等处理器不同的锁相环(PLL)设计，因此，MPC866 处理器的大部分初始化代码可以直接应用到 MPC852 处理器上。

HHPPC852T 开发板的硬件设计与 TQM866M 标准板有很多相似的地方，主要包括硬件复位向量、存储器(包括 Flash 和 RAM)地址分配、存储器片选设置以及 CPM 通信通道的分配等。

5　结束语

通过本文的分析和实践，已经能够成功地将 U-Boot 移植到 MPC852T 目标板上了，为基于平台的 U-Boot 移植提供了一个成功范例，同时也为移植 U-Boot 到其他目标板积累了丰富的实践经验。尽管 U-Boot 移植的具体过程千差万别，但是总体上来说，必须围绕以下几个要点，包括：U-Boot 移植参考板；CPU 寄存器参数设置；串口调试；与 Flash 相关的寄存器 BRx, ORx 的参数设置；SDRAM 的驱动等。应尽量参考已经可以在目标板上成功运行的 Bootloader，从中可以获得很多有价值的资料，从而降低移植难度，缩短移植调试的时间。另外，在移植过程中，必须通过各种渠道获取帮助[4]。

如果在自主设计的目标板上移植 U-Boot，那么除了考虑上述软件因素以外，还需要借助硬件测试工具排查目标板硬件可能存在的问题，如原理设计、PCB 布线、元件好坏等。因此，在移植过程中，需要对移植的问题进行综合考虑，积极与硬件工程师配合，敏锐地区分硬件问题和软件问题。

参考文献

[1] 詹荣开. 嵌入式 Bootloader 技术内幕[EB/OL]. (2003-12-22). http://www.900.ibm.com/developerWorks/cn/linux/l-btloader/index.html.

[2] 李　毅, 李连云, 张伟宏, 等. Bootloader 面向不同结构 Flash 的实现[J]. 计算机工程, 2008, 34(4): 82-83.

[3] 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): 78-81.

[4] Denk W. The DENX U-Boot and Linux Guide(DULG) for TQM8xxL[EB/OL]. (2004-11-18). http://www.denx.de/wiki/bin/view/DULG/Manual.

编辑　顾逸斐

(上接第 266 页)

参考文献

[1] Bao Tie, Liu Shufen. A Method for Network Data Collection and Processing in the Pervasive Computing Environment[C]//Proc. of International Symposium on Pervasive Computing and Applications. [S. l.]: IEEE Press, 2006: 599-603.

[2] Kim Do-Hyeon, Kang Kyung-Woo. Design and Implementation of Integrated Information System for Monitoring Resources in Grid Computing[C]//Proc. of the 10th International Conference on Computer Supported Cooperative Work in Design. Nanjing, China: IEEE Press, 2006.

[3] 张淑英, 刘淑芬, 包　铁. 一种基于机群的分布式数据采集系统[J]. 计算机工程, 2006, 32(14): 46-48.

[4] Song Hengjie, Shen Zhiqi, Miao Chuyan. The Multi-Agent Data Collection in HLA-based Simulation System[C]//Proc. of the 21st International Workshop on Principles of Advanced and Distributed Simulation. San Diego, California, USA: IEEE Computer Society, 2007.

[5] 缪嘉嘉, 邓　苏, 刘青宝. ETL 综述[J]. 计算机工程, 2004, 30(3): 4-5.

编辑　张正兴