基于 U-boot 的 USB 接口系统更新方案设计

余柳冰,高明煜*
(杭州电子科技大学 电子信息学院,浙江 杭州 310018)

摘要:为了在保证较低成本的基础上提高嵌入式设备的系统更新速度,分析了串口、网络以及 USB 接口这 3 种可选方案的优、缺点,研究并设计了基于通用加载器(U-boot)的 USB 接口系统更新方案。具体介绍了 U-boot 下 USB 设备端口驱动的实现,着重分析了 U-boot 下中断向量表的搬移过程,并在 AT91SAM9261 处理器平台上实现了基于 U-boot 的系统下载更新功能。实验结果证明,该方案可以在简单的硬件配置上达到 500 KB/s 左右的更新速度。

关键词:通用加载器;中断向量表;系统更新
中图分类号: TP392　　**文献标志码**:A　　**文章编号**:1001 - 4551(2011)09 - 1109 - 04

Design of system update based on U-boot through USB interface

YU Liu-bing , GAO Ming-yu
(School of Electronics Information, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: In order to improve the system-updating speed of embedded devices on the basis of low cost, both advantages and disadvantages of serial port, network and USB interface were analyzed. A system-updating method through USB interface based on universal boot loader(U-boot) was implemented . The implementation of USB device driver in U-boot was introduced, the movement of the interrupt vector table was analyzed, and the updating function was successfully accomplished on AT91SAM9261 processor. The experimental results show that the fast updating speed about 500 KB/s has been achieved with simple hardware configuration in practice.

Key words: universal boot loader(U-boot); interrupt vector table; system-updating

0　引　言

操作系统的下载更新速度是嵌入式设备开发过程中需要考虑的一个关键因素。系统更新不仅要考虑速度问题,还要考虑到实现系统更新所需硬件接口的复杂程度。目前有 3 种可选的方案可以实现嵌入式操作系统的下载更新[1]:通过串口、网络以及 USB 接口。串口的缺点是速度太慢,不论是调试阶段还是设备更新阶段,都不是理想的传输方案;通过网络接口传输是嵌入式设备调试阶段中通常采用的一种文件传输方式,它的优点是传输速度很快,缺点是网卡芯片的成本很高,而且网络接口会占用较大的物理空间;相比之下,使用 USB 接口就简单多了,现在的嵌入式处理器都内带 USB 设备接口,该接口在系统更新过程中可作为传输操作系统文件的通道,在设备使用过程中则作为正常的 USB 接口,进行普通文件的传输。该方案可以在保证较快的下载速度基础上有效地减少设备成本。

U-boot 是遵循 GPL 条款的开放源代码项目[2],它支持多种嵌入式操作系统内核,支持多个系列处理器,具有较高的可靠性和稳定性,而且包含丰富的设备驱动源码。一般情况下 U-boot 在嵌入式设备中的主要功能是加载并引导操作系统,而设备的系统更新则在上层应用程序中予以实现。该种方式的不足之处在于,由上层应用配合系统的更新会使系统的下载更新时间有所增加,并且在实现相同更新功能的基础上,在

收稿日期:2011 - 03 - 01
作者简介:余柳冰(1985 -),男,浙江绍兴人,主要从事嵌入式系统驱动方面的研究与开发. E-mail:2673986@163.com
通信联系人:高明煜,男,教授,硕士生导师. E-mail: mackgao@hdu.edu.cn

上层应用中开发更新程序的周期较在 U-boot 下开发更长。

为了缩短嵌入式设备的开发时间且提高系统的下载更新速度,本研究考虑在 U-boot 中添加系统更新功能。

1 USB 设备端口及其中断过程的介绍

AT91SAM9261 处理器的 USB 设备端口(USB Device Port, UDP)适用于通用串行总线全速设备规范[3]。UDP 控制器提供了 6 个端点,分别是端点 0 ~ 5。其中端点 0 和 3 主要用做控制端点外,其他端点均可配置为中断、等时或批传输方式,批传输包的长度为 8、64 或 256 (单位:Bytes)。UDP 控制器的工作原理为:当 UDP 控制器从 USB 总线检测到主机发出的某一传输请求时,UDP 控制器将相应标志置位。如果配置为中断方式,则向中断控制器请求中断。软件通过访问 UDP 控制器的状态寄存器和数据寄存器获得与此次传输有关的各种参数,并根据参数对 UDP 控制器的控制寄存器和数据寄存器进行相应的操作,以完成主机的传输请求。

由于处理器支持向量化中断功能,使得中断处理更加迅速。当中断产生时,装载在地址 0x18 处的指令被执行,将中断返回地址及其他通用寄存器作入栈处理,而后读取中断向量寄存器中保存的中断函数入口地址,执行相应的中断函数代码。中断处理完成后,写中断结束命令寄存器以表示当前中断已完成。最后进行出栈操作,并将被中断的函数地址返回给程序计数器,整个中断过程完成。

2 主机对 USB 设备的枚举过程介绍

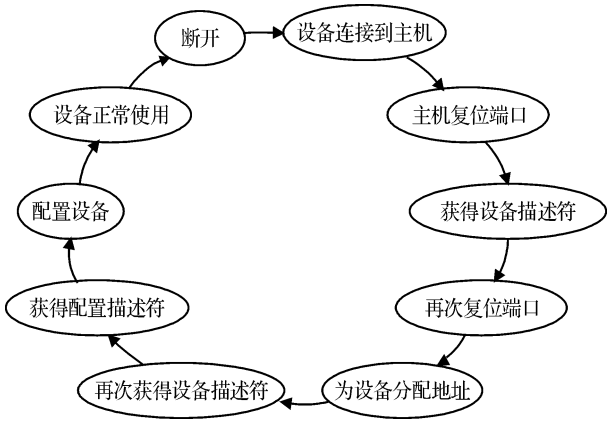

图 1 枚举过程状态图

主机在与 USB 设备进行数据传输之前,必须首先对 USB 设备进行枚举[4]。主机一开始通过端点 0 地址与设备进行通讯,然后通过给设备分配的地址与设备通讯。对设备进行配置后,枚举过程结束,此时设备才可正常使用。枚举过程中,若没有 USB 总线活动,则设备将挂起等待 USB 总线活动。枚举过程状态图如图 1 所示。

3 U-boot 下 USB 设备端驱动的具体实现

笔者采用的 U-boot 版本是 U-boot-2009.06。该版本对各类型处理器的 USB 设备端口驱动的支持较弱,只支持 omap1510 以及 pxa27x 系列的处理器。目前版本的 Linux 内核已经实现了 AT91SAM9261 处理器的 USB 设备端口驱动,因而可以借鉴 Linux 内核的代码并做一些修改。

3.1 中断向量表的搬移

启动代码的运行过程,如图 2 所示,AT91SAM9261 处理器复位后,本研究根据特定引脚的设置,选择从内部固化 ROM 启动 ,ROM 中的代码开始运行,初始化处理器和必要的外设。然后开始从 Data Flash 的 0 地址检索合法的启动程序,也就是 Bootstrap,如果合理的 Bootstrap 存在,ROM 程序将其复制到内部 SRAM 并跳转到 Bootstrap 运行。Bootstrap 将初始化 Data Flash 与 SDRAM 等一些设备,然后从 Data Flash 特定位置将 U-boot 代码复制到 SDRAM 的指定位置,然后跳转到该指定位置开始运行。

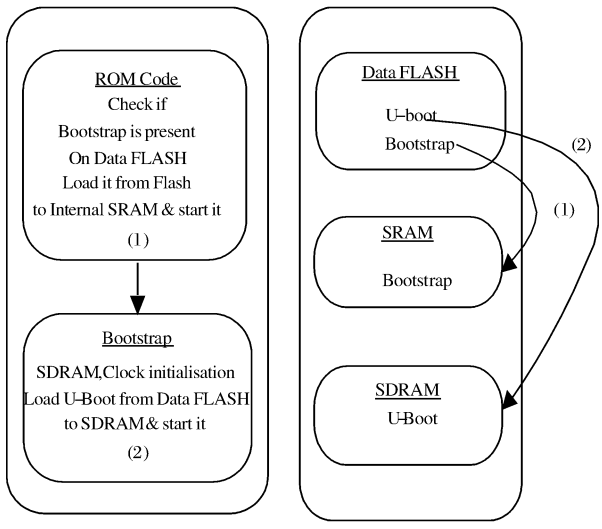

图 2 启动代码的运行过程

驱动开发过程中的第一个重要问题是如何在起始地址为 0 处建立中断向量表（interrupt vector table，IVT）[5]。处理器默认从内部固化的 ROM 启动，但是在该地址并不存在中断向量表，并且无法修改内部 ROM 的内容，因而只能通过地址的重映射来实现[6]。重映射允许用户映射第一个内部 SRAM 存储器到 0x0。具体过程如图 3 所示，U-boot 在 SDRAM 中运行后，先将位于 U-boot 首部的中断向量表以及总中断函数入口处理部分（主要是读取中断向量寄存器）拷贝到重映射之前首地址为 0x300000 的内部 SRAM 中，然后进行重映射操作，使内部 SRAM 映射到 0 地址，这样就将中断向量表搬到了 0 地址。具体的实现代码如下[7]：

```
ldr   r0, =0x23f00000      //r0 保存 U-boot 首地址
ldr   r1, =0x300000        //r1 保存重映射之前的内部 sram 首地址
ldr   r3, =0x23f00200
sub   r2, r3, r0           //r2 保存中断向量表以及总中断函数入口处理部分大小
add   r2, r2,r0            //r2 保存代码源结束地址
copy_loop:                 //进行拷贝操作
ldmia   r0!, {r3-r10}
stmia   r1!, {r3-r10}
cmp   r0, r2               //比较 r0,r2 判断是否拷贝完成
ble   copy_loop
ldr r0, =BOARD_RemapRam//BOARD_RemapRam 为实现重映射的函数
blx r0 //进行重映射操作，完成后返回
```

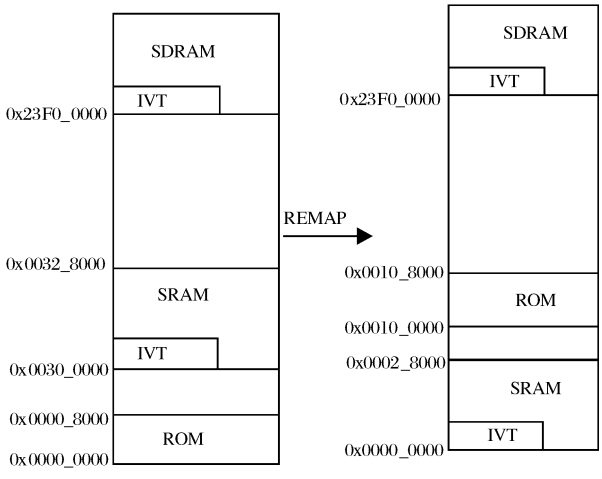

图 3　重映射实现过程

3.2　USB 设备的初始化设置

默认的 U-boot 源代码中没有开启中断功能来处理外设的数据传输，而是通过查询的方式来实现的。要使能中断功能，需要调用 Enable_Interrupts 函数，该函数位于/lib_arm/interrupts. c 文件中，对所有 ARM 架构的处理器都适用。值得注意的是调用该函数前应初始化相应的寄存器以防止误中断的产生。USB 设备的初始化入口函数为 Drv_Usb_Init，其中对设备配置、接口以及端点结构体的初始化参考 U-boot 源代码中的 usbtty. c 文件。以下为 Drv_Usb_Init 调用的子函数以及实现的功能[8]：

Usb_Clock_Init	使能外设及 USB 时钟
Pio_InitInterrupts	配置并使能并行 I/O 口中断，用于检测设备是否连接到主机
Usbd_Init	初始化并使能 UDP 中断
Vbus_Configure	检测设备是否到连接主机
Usb_Init_Device	初始化端口配置及接口结构体
Usb_Init_Endpoints	初始化所有端点（包括控制端点 0）

3.3　USB 设备的中断函数的实现

由于 U-boot 的 USB 设备驱动不支持 AT91SAM9261 处理器，笔者借鉴了 Linux-2. 6. 24 版本内核中的源代码文件 at91_udc. c，并且作了一部分的修改[9]。at91 _udc. c 文件主要提供 USBD _ InterruptHandler 函数，该函数处理 UDP 控制器接收到的所有中断，包括复位、挂起、恢复以及各端点的中断，以完成主机对设备的枚举过程及端点的批量传输操作。对控制端点 0 的通用操作（如 get descriptor）则由 ep0. c 及 core. c 实现，这两个函数由 U-boot 源代码提供，与处理器类型无关，基本不作修改。

4　调试与验证

4.1　调试过程

调试软件采用 ARM 公司调试器 RVDS，RVDS 是一款性能优越的调试器，几乎支持所有基于 ARM7 和 ARM9 内核的处理器，本研究在调试过程中通过 RVDS 的代码编译器阅读并且修改汇编代码，用 RVDS 调试器进行断点和单步调试。仿真器采用的是 segger 公司的 J-LINK，该工具价格适中，并且支持大部分的 ARM 处理器，所需的驱动可到 segger 官方网站上下载。整个调试步骤如下：

（1）首先在 PC 机上安装调试软件 RVDS；

(2)下载并安装 J-LINK 驱动软件,安装完成后需要将 J-LINK 整合到 RVDS 中,整合过程参考微控电子提供的使用手册,以后每次启动 RVDS,J-LINK 软件也会随之启动;

(3)将 U-boot 源代码文件夹通过 Linux 操作系统提供的 samba 服务映射为一个 Windows 操作系统下的一个本地盘符,以便在调试的时候定位代码来进行断点和单步调试;

(4)在 Linux 操作系统下修改并交叉编译 U-boot 源代码,生成 ELF 格式的二进制调试文件;

(5)在 RVDS 通过 J-LINK 将 elf 文件下载到目标板的 SDRAM 上,运行一小段时间后由 J-LINK 控制处理器让其暂停,而后由用户进行单步调试或断点调试;

(6)重复(4)、(5)两个步骤,直到实现所需的功能;

(7)最后将后缀为 bin 的 U-boot 二进制文件烧写到外部 FLASH 中。

上位机(安装 Linux 操作系统)采用网友编写的软件 dnw_linux,运行之前需要加载一个 USB 驱动。由于 dnw_linux 软件只支持三星公司设定的设备的厂商号及产品号,需要修改 U-boot 中定义设备厂商以及产品号的相关结构体,以使 dnw_linux 软件能正确识别出 USB 设备。

4.2 实验验证

实验主要分为两个部分。第一部分是通过串口、网络及 USB 接口传输同一文件到处理器的 SDRAM 中,其大小为 9.5 MB,3 种不同下载方式的运行情况如表 1 所示。

表 1 普通文件下载对比表

方式	所需时间/s	出错率
串口	2 220	有时
网络	7	无
USB 接口	21	无

第二部分是进行实际的操作系统更新,通过串口、网络及 USB 接口下载操作系统内核到设备的 DATAFLASH 中[10],其大小为 1.5 MB,3 种不同下载方式的运行情况如表 2 所示。

由实验可知,USB 接口的下载速度虽不及网络,但也达到了 500 KB/s 左右,并且具有很好的稳定性,因而通过 USB 接口下载更新操作系统是一个不错的选择。

表 2 操作系统更新对比表

方式	所需时间/s	出错率
串口	420	有时
网络	2	无
USB 接口	4	无

5 结束语

本研究介绍了 U-boot 下 USB 设备端驱动的整个开发过程,实现了中断向量表的搬移,并将驱动程序整合到 U-boot 代码下,最终实现了设备的操作系统更新功能。经过实测,下载一个 9.5 MB 左右的文件需要 21 s 左右,而下载更新 1.5 MB 左右的操作系统内核仅需 4 s,因此本研究所介绍的基于 U-boot 的 USB 更新方法具有较高的实用价值,而且在一些条件比较苛刻的调试场合(如无法使用网络进行调试),也可通过 USB 接口进行操作系统内核的调试。

参考文献(References):

[1] 许杨,胡晨,姚国良. Bootloader 下 USB 接口镜像下载方案研究与实现[J]. 电子器件,2007,30(2):1-2.

[2] 韦东山. 嵌入式 Linux 应用开发完全手册[M]. 北京:人民邮电出版社, 2008.

[3] 微控电子. AT91SAM9261 中文数据手册[EB/OL]. [2007-03-13]. http://www. mcuzone. com.

[4] 周立功. USB 2.0 与 OTG 规范及开发指南[M]. 北京:北京航空航天大学出版社, 2004.

[5] ATMEL. AT91 C-startup[EB/OL]. [2006-09-21]. http://www. at91. com.

[6] 裴科,张刚,靳荣浩. 具有多重下载接口的 Bootloader 设计[J]. 计算机应用研究,2007,24(12):1-2.

[7] 杜春雷. ARM 体系结构与编程[M]. 北京: 清华大学出版社, 2003.

[8] REEK K A. C 和指针[M]. 北京:人民邮电出版社, 2008.

[9] CORBET J. LINUX 设备驱动程序[M]. 北京:中国电力出版社, 2008.

[10] 施文灶,王平,黄晞,等. 基于 USB 的便携式设备固件升级程序的设计[J]. 电子设计工程,2009,17(11):1-2.

[编辑:李 辉]