Bootrom 功能改进经验谈

上海贝尔阿尔卡特有限公司 FCG 部(201206)　王　江

摘　要：以 VxWorks 嵌入式实时操作系统为例，介绍了改进启动代码(Bootrom)功能的四点经验：(1)具备故障保护功能的 Bootrom 的自我在线更新；(2)增加重启功能；(3)自动运行应用软件；(4)多种应用软件的选择。

关键词：Bootrom　嵌入式实时操作系统　FTP

VxWorks 是美国 Wind River System 公司于 1983 年设计开发的一个运行在目标机上的高性能、可裁减的嵌入式实时操作系统(RTOS)。它是一种功能强大而且比较复杂的操作系统，包括了进程管理、存储管理、设备管理、文件系统管理、网络协议及系统应用等几部分。VxWorks 为程序员提供了高效的实时多任务调度、中断管理、实时的系统资源以及实时的任务间通信。其核心功能主要有微内核、任务间通信机制、网络支持、文件系统和 I/O 管理、POSIX 标准实时扩展以及 C++等其他标准支持。在各种 CPU 平台上提供了统一的编程接口和一致的运行特性，尽可能地屏蔽不同 CPU 之间的底层差异。应用程序员可以将尽可能多的精力放在应用程序本身，而不必关心系统资源的管理。VxWorks 以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通信、弹道制导、飞机导航等。

设计师通常利用 VxWorks 开发 Bootrom 代码。Bootrom 相当于 PC 机中的 BIOS，它完成对加载过程中所需设备的初始化及驱动；然后，通过某种可选择的通信手段(如网口、串口)，将 VxWorks 内核加载。VxWorks 内核相当于 PC 机上的操作系统，如 Linux、Windows。此外，Bootrom 还提供了一些辅助功能，如地址内容查看、地址内容修改和 Bootrom 菜单显示信息控制等功能。但是，在实际应用中，这些功能不够丰富，便利性不足。本文介绍了几点笔者在工作中行之有效的改进 Bootrom 功能的经验。

1 具备故障保护功能的 Bootrom 的自我在线更新

Bootrom 一般以二进制文件的方式保存在非易失性存储介质，例如 Flash、CF 卡、EPROM 中。通常需要仿真器、烧结器或 JTAG 工具等才能将代码烧入这类介质，在实际应用中非常不便。以 Flash 为例介绍如何在线更新 Bootrom。

首先，为了做到故障保护，防止在更新过程中发生更新文件出错、断电等灾难性故障，Bootrom 的数据必须在更新之前备份在 Flash 的另一区域。因此，Flash 中必须存在两片物理区域，暂且命名为 PA 和 PB，每片 512KB(假设 Bootrom 文件小于 512KB)。两片物理区域的起始地址可以互换，其中一片区域的起始地址必须为系统的上电启动地址，这个地址因处理器而异，例如，PowerPC 体系结构的处理器的启动地址一般是 0xfff00100，而 MIPS 体系结构的处理器的启动地址是 0xbfc00000。在逻辑上，以启动地址开始的区域为主 Boot 区，另一片区域为备 Boot 区。

为了实现地址互换，在 CPU 模块和 Flash 芯片之间，增加了一片 CPLD(可编程逻辑器件)和一个跳线或拨号开关，开关信号 JP 输入到 CPLD。CPU 访问 Flash 的地址信号和片选信号经过 CPLD 进行地址互换，再到 Flash 芯片。如果不跳线(默认情况)，JP 信号为高，则 PA 的起始地址为启动地址，PA 为主 Boot 区，PB 为备 Boot 区；否则，JP 信号为低，PB 的起始地址为启动地址，PB 为主 Boot 区，PA 为备 Boot 区。默认情况下，PA 为主 Boot 区。如图 1。

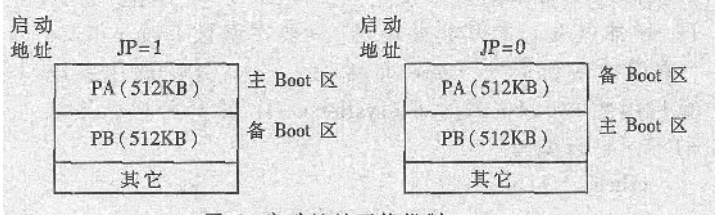

图 1　启动地址互换机制

如果系统中有两片 Flash 芯片，可以不用 CPLD，选用另一种地址互换方式，如图 2。两个二选一逻辑实现两片 Flash 片选信号(CS)的互换，从而实现了地址的互换。

地址互换机制屏蔽了 PA 和 PB 实际物理位置的差别，给底层软件提供了一个统一的接口，带来的好处不言而喻。Bootrom 数据总是从主 Boot 区备份到备 Boot 区，系统总是从主 Boot 区启动。

其次，编写 Flash 擦写驱动函数。Flash 是一种读写非对称器件，读数据与普通器件一样，比较简单；而写必须根据芯片厂商提供的算法，先擦除欲写入数据的区域，以扇区(Sector)为单位，然后写入数据。函数 flashUpdate (char *sourceAddr, char *destAddr, int Length)集成了擦除

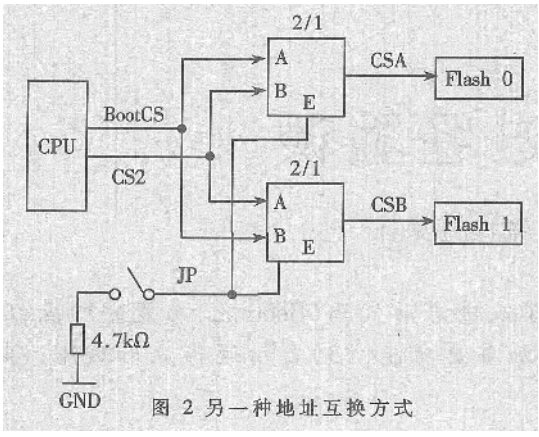

图 2 另一种地址互换方式

和写入两个步骤。

再次，在文件 bootConfig.c 的 Bootrom 命令操作接口函数 bootCmdLoop()中增加新命令“u”（update 的缩写）。

```
switch (*(pLine++)){
……
    case 'u':          /* 更新 Bootrom */
        ……             /* 下载更新代码并更新 */
    break;
……
}
```

“u”命令的实现流程如图 3。除了 flashUpdate()之外，其它函数都由 VxWorks 本身提供。如果更新失败，则改变 JP 设置，从备份 Bootrom 启动，再次更新 Bootrom 或运行应用软件。

2 增加重启功能

VxWorks 的 Bootrom 用户命令缺少重启功能，给调试和实际应用带来诸多不便。与 Bootrom 在线更新方法相同，增加这项功能其实并不难，只要在函数 bootCmdLoop()中增加新命令“r”（reset 的缩写），在命令中调用系统重启函数 sysReboot()即可。sysReboot()由设计师根据系统的不同自行编写。

```
switch (*(pLine++)){
……
    case 'r':                  /* 重启系统 */
        sysReboot();
    break;
……
}
```

举一反三，还可以根据需要在 Bootrom 中增加许多自己的命令。完成之后，不要忘了在 Bootrom 命令解释函数 bootHelp()中添加新增命令的用途、用法等帮助信息。

3 自动运行应用软件

在实际应用中，一般将上层软件和 VxWorks 映象链接在一起，暂且称之为应用程序。默认情况下，当 Bootrom 运行后，它会倒计时 7s，然后从启动参数

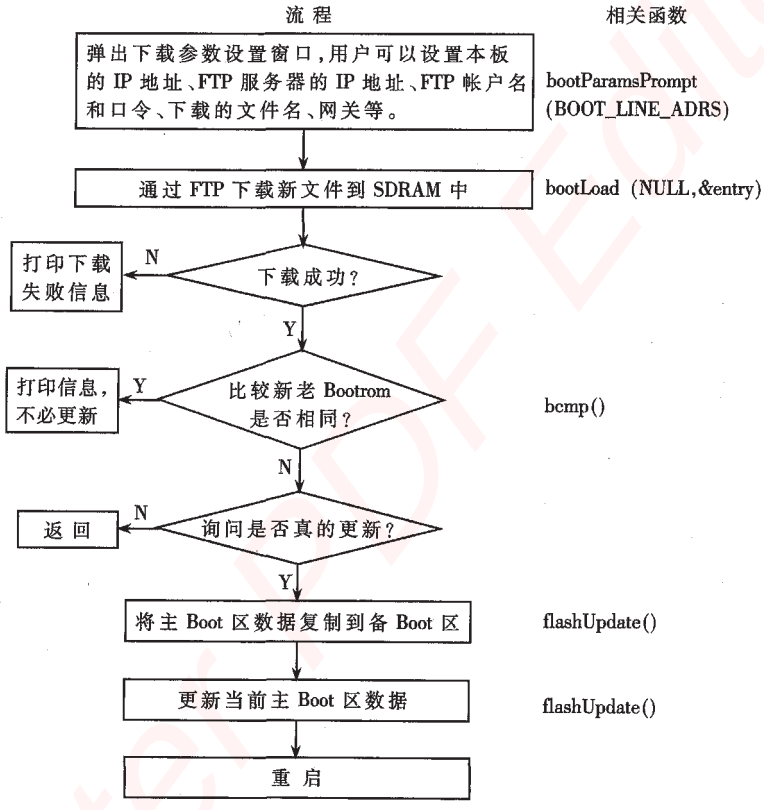

图 3 “u”命令实现流程

（BOOT_LINE）指定的路径下载应用程序。但是，因为 Bootrom 提供了启动参数修改命令，启动参数可能由于某些原因发生更改。例如调试时需要从 FTP 服务器下载，或者疏忽好奇等，给下次应用带来不少麻烦，很可能出现应用程序下载失败等故障。

为了保证自动启动总是从固定路径下载应用程序，而不受启动参数的影响，需要修改文件 bootConfig.c 中的自动启动函数 autoboot()。例如，自动启动总是从 CF 卡读取应用程序。

```
if (bytesRead == 0) {          /* 倒计时期间没有输入 */
    LoadApplicationImage(); /* 复制应用程序映象到起
                               始地址为 Entry 的内存中 */
    go ((FUNCPTR ) Entry); /* 从 Entry 地址开始运行 */
}
else{
    ……
}
```

Entry 是应用程序的入口地址，VxWorks 系统中该值是 RAM_LOW_ADRS。LoadApplicationImage()函数将应用程序映象从 CF 卡复制到起始地址为 Entry 的内存中，然后将 PC 指针指向该地址，启动系统。如果在倒计时期间有字符输入，则进入 Bootrom 的命令界面，用户可以修改启动参数，然后按“@”从启动参数指定的路径下载程序，而不影响调试，但是更改后的启动参数不会影响下次自动启动的下载路径。

4 多种应用软件的选择

在通信领域，许多产品的硬件平台其实相同，区别在于应用软件。不同的软件塑造出了一个个特色鲜明的产品。如果能够将不同的软件集成到一个硬件平台上，由用户在启动时选择，那么这个产品就拥有一机多能的特性，大大增强了产品的市场竞争力。

退而言之，即使硬件平台只适合于一种应用软件，将故障检测程序作为另一种应用软件，会带给产品测试和现场维修人员诸多的方便。

参考以上三点经验，在 Bootrom 中增加这项功能并非难事。以两个应用软件为例，在函数 bootCmdLoop()中增加新命令“o”(other 的缩写)，如下所示。

```
switch (*(pLine++)){
……
    case 'o':                          /* 启动另一应用软件 */
        LoadAnotherApplicationImage();  /* 复制另一应用程序映象到起始地址为 Entry 的内存中 */
        AutoSystemVersionSet();  /* 询问用户是否将另一应用软件设置为默认启动 */
        go ((FUNCPTR ) Entry ); /* 从 Entry 地址开始运行 */
    break;
……
}
```

在 Bootrom 代码中定义一个变量 defaultVersion，用于记录谁是默认的启动软件，该变量保存在 Flash 等非易失媒质中。上电后，如果在倒计时期间没有任何输入，系统自动启动 defaultVersion 指定的应用软件；否则，用户进入命令界面。如果键入“o”命令，Bootrom 调用函数 LoadAnotherApplicationImage () 将另一应用程序映象复制到起始地址为 Entry 的内存区域，然后询问用户是否希望将另一应用程序设置为默认启动；如果用户回答“是”，更改 defaultVersion 值；否则，defaultVersion 值不变。最后运行当前选定的应用软件。

上述的几点经验大部分是修改文件 bootConfig.c 的代码。因为 bootConfig.c 是一个公用文件，为了不影响其它底层软件的开发，建议将它复制到当前 Bootrom 开发目录下，并且在 Makefile 文件中增加定义：BOOTCONFIG=bootConfig.c。

嵌入式系统开发需要经常开辟新的思路，一些微小的简单改动通常能够带给产品新的功能和特征。启动代码是嵌入式系统的重要组成部分。笔者的以上四点经验在 PowerPC 处理器 (MPC8260) 平台和 MIPS 处理器 (RM7000A)平台上得到了验证，希望对其它系统的启动代码的开发有所借鉴和启迪。

参考文献

1 Windriver Inc. VxWorks BSP Developer's Guide, 2002
2 Windriver Inc. VxWorks Programmer's Guide, 2002

(收稿日期：2004-04-14)

(上接第 3 页)

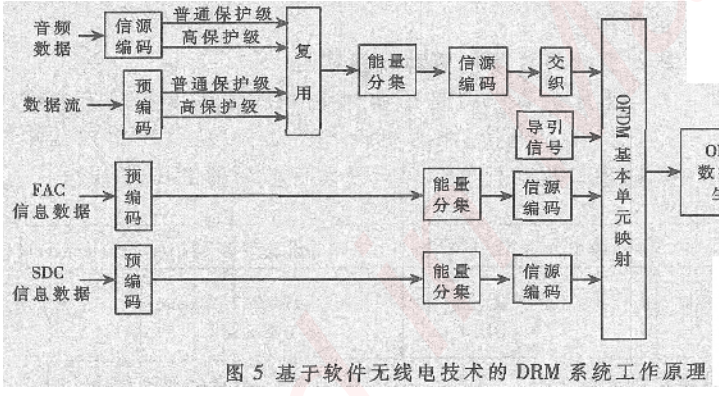

图 5 基于软件无线电技术的 DRM 系统工作原理

织、数字基带的 OFDM 映射部分的功能将在数字编码与调制子系统中利用计算机的处理器、DSP 处理器以及专用芯片等通过软件编程来实现。而无线射频信号的生成、稳定载波的产生等模拟处理功能将在模拟处理子系统中通过 DDS、I、Q 调制器等技术或专用器件实现。

数字广播领域市场广阔，具有很好的发展空间，目前世界各个主要发达国家都在此领域投入了相当的人力、物力、财力。我国在这一领域的研究水平与国际同步，更不能放弃这一优势。

软件无线电技术一经提出就被认为是无线电领域的一场革命，近年来“软件无线电”的思想已经渗透进了仪器仪表、自动控制、信号处理等诸多领域。我国在这一领域的研究也取得了显著的成果。

将软件无线电技术与数字广播技术结合在一起，对于数字广播技术发展和数字广播设备的推广具有巨大的推动作用。

参考文献

1 何晶莹.调幅广播的新技术革命与世界数字广播(DRM)集团[J]，广播与电视技术，2002 年第 4 期，p61-65
2 杨小牛，楼才义，徐建良[M].软件无线电原理与应用.北京：电子工业出版社.2001

(收稿日期：2004-6-20)

熊猫电子集团将全面参与数字高端产品的竞争

熊猫电子集团近日在北京举行了数字高端产品新闻发布会。从产品阵容看，有液晶电视、等离子电视、DLP 光显背投和 CRT 数字高清电视，四大类十余款产品。熊猫家电产业集团正式宣布，熊猫将全面参与数字高端产品的竞争。

2005~2006 年是熊猫“数字互动年”。熊猫将推出与互联网完美结合，与家庭数码产品完全兼容的数字互动产品。

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

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

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

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

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

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