文章编号:1001 - 9081(2005)05 - 1055 - 03

uClinux 在 S698 处理器上的移植研究与实现

占文静, 张 凯
(北京理工大学 计算机科学与工程系,北京 100081)
(icejingzhan@ sohu. com)

摘 要:研究了如何在自主知识产权的"中国芯"S698 上移植 uClinux 操作系统。选择对 SPARC NOMMU 支持的内核版本 uClinux2.0.x,在简单介绍 S698 硬件结构后,就开发环境、引导程序、操作系统内核的移植、romfs 文件系统、操作系统映像的构建等方面进行研究分析,并给出实现方法步骤,该移植在 S698 处理器应用开发板上实现。

关键词:S698; SPARC; uClinux
中图分类号: TP316.2 **文献标识码**:A

Research on transplant of uClinux in S698 microprocessor

ZHAN Wen-jing, ZHANG Kai
(*Department of Computer Science and Engineering, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: The method of transplanting open source uClinux to S698 microprocessor, which was made in China and had its own intellectual property, was studied. After the introduction of S698 architecture, the development environment was built, bootstrap, romfs, OS kernel transplant and construction were analyzed, and the solution to these problems was brought forward. The transplant was implemented on S698 Application Development Board.

Key words: S698; SPARC; uClinux

1 S698 处理器

SAILING S698 处理器芯片是继"方舟"、"龙芯"、"众志"之后,又一具有高端技术和自主知识产权的"中国芯",主要用于航天领域。

S698 处理器为 SPARC V8 系列无 MMU 的 32 位 RISC 嵌入式处理器,其内部配置了遵循 SPARC V8 标准的 32 位整形数运算单元(IU)、符合 ANSI/IEEE 754 标准的 64 位浮点运算器(FPU)、AMBA 总线架构、PCI 控制器接口、片上外设(UART、定时器、中断管理器、I/O、看门狗等)、片上调试支持单元(DSU),以及存储器控制接口等。S698 采用 AMBA 总线作为片内系统架构总线。

S698 处理器采用 Harvard 结构,地址数据总线分开,分别连接相互独立的缓存(CACHE)控制器;内部配置了双时钟机制,即 CPU 时钟(CPUCLK)和片内总线时钟(SYSCLK);CPU 内部指令实行单指令发射流水线,具备五级流水(PIPELINE)[3]。

S698 处理器内部结构如图 1 所示。

作者使用 S698 处理器应用开发板作为移植实现平台,S698 处理器应用开发板上有 S698 处理器、CPU 时钟晶振和 PCI 时钟晶振、多时钟及倍频配置模块、存储器模块(包括 SDRAM、SRAM 和 PROM 等)、TAP 调试接口(COM3)及串行通信接口(COM1 和 COM2)、PCI 连接器等[1]。

2 移植

选择对 SPARC NOMMU 支持的内核版本 uClinux 2.0.x 进行移植。移植工作主要从开发环境、引导代码、操作系统内核的移植、romfs 文件系统、操作系统映像的构建五个方面展开。

2.1 开发环境的建立

1) 开发工具链的选择。S698 遵循 SPARC 标准,因此可以选择 sparc 系列开发工具如 Sparc-elf-gcc、Sparc-elf-ld、Sparc-elf-objdump 等。

2) 调试方式。利用 S698 处理器内部集成的调试支持单元(DSU)进行片上调试。

S698 处理器内部集成了 DSU 及 DSU 串行通信连接

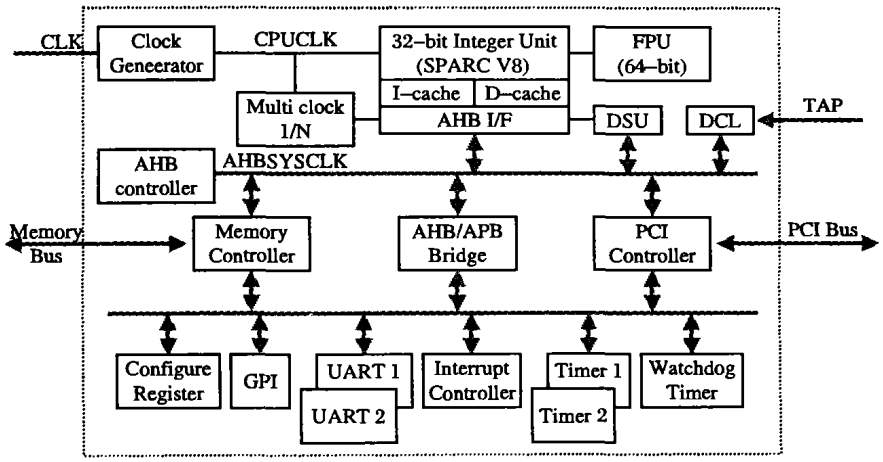

图 1 S698 处理器内部结构框架

收稿日期:2004 - 10 - 19;**修订日期**:2004 - 12 - 29

作者简介:占文静(1981 -),女,江西九江人,硕士研究生,主要研究方向:计算机嵌入式系统; 张凯(1949 -),女,北京人,副教授,主要研究方向:计算机系统结构,计算机嵌入式系统.

(DCL),用来进行在线调试软件,即 DSU 调试模式。DCL 的对外接口(TAP)为 RS232 标准。DSU 及 DCL 内部实现了简单的通信协议。在 DSU 调试模式下,用户可以通过 TAP 接口查看处理器内部的所有资源,包括 IU 内部寄存器、系统寄存器等,同时还可以读写目标板上的存储器。

3) 内核的下载和烧写。Dsumon 软件能通过 DSU 将内核从主机下载到目标板 RAM 中并运行,但调试好的内核需要烧写到 Flash(即图 1 中的 PROM)中,而 Dsumon 软件不提供写 Flash 功能,因此要实现 Flash 烧写。其实现包括两过程:

· 从串口下载可执行代码(以文件方式)到 RAM 中;

· 把 RAM 中的可执行代码烧写到 Flash ROM 中。

烧写程序主要包括启动部分、串口驱动、Flash 驱动和时钟驱动等。其中启动部分包括:包括 Trap 矢量表初始化、硬件初始化和必要的 Trap 处理。Rrap 矢量表初始化时让各未用 trap 指向同一个处理函数,以防止错误中断引起系统混乱。

2.2 引导程序

引导程序(也称 Bootstrap)就是在操作系统内核运行之前运行的一段小程序,它在复位后首先被执行。通过这段小程序,我们可以初始化硬件设备、建立内存空间的映射图,从而将系统的软硬件环境带到一个合适的状态,以便为最终调用操作系统内核准备好正确的环境。通常引导程序严重地依赖于硬件,每种 CPU 体系结构都有不同的启动代码。此外,引导程序实际上也依赖于具体的嵌入式板级设备的配置。

对于 S698 处理器应用开发板,引导程序主要完成以下几项工作:

1) Trap 矢量表初始化。

2) 硬件初始化。

3) 将操作系统内核中的.data 数据段从 Flash 上拷贝到 RAM 中。

4) 软件初始化:设置堆栈指针并将.bbs 段清零,建立 C 语言的运行环境。

5) 改变 PC 值,使得 CPU 执行真正的操作系统内核。

第 2 步主要初始化处理器的一些控制/状态寄存器、时钟分割因子、串口、DRAM、SRAM、看门狗等这些内核运行必须用到的设备。

解释第 3 步,uClinux 内核加载方式有两种:在 Flash 上直接从某个地址开始逐句运行或加载到内存中运行。如采用后者需将整个操作系统内核从 Flash 拷贝到 RAM 中,如果是压缩格式的内核,还要先解压。作者采用在 Flash 上直接运行内核的方式,所以只要拷贝.data 数据段。

引导程序形式如下:

```
set   0x0000083, %g2
                    ! 初始化存储器配置1,PROM 数据宽度设为 8 位
st    %g2, [%g1 + MCFG1]
set   UART_SCALER_VAL, %g2
                    ! 初始化串口串口波特率,其值要用公式计算
st    %g2, [%g1 + USCAL0]
...
Call ...                    !进入 C 语言环境,调用 start_kernel()
```

2.3 操作系统内核的移植

引导程序执行完毕后,CPU 就开始执行真正的操作系统内核 start_kernel()函数。此函数调用了多个子函数,并在这些子函数中对系统中的各种硬件进行初始化。下面按照操作系统启动时对各种硬件初始化的顺序并结合 S698 处理器来介绍操作系统源码中哪些文件和函数需要移植和改写。

2.3.1 Trap 和中断

操作系统硬件初始化时,会调用 trap_init()和 init_IRQ()。这两个函数的功能都是对中断处理进行初始化,不同之处在于 trap_init()主要负责 CPU 中断向量表的初始化和各种中断共用的中断服务程序(如保存中断现场、获得当前中断类型等例行程序)的初始化,而 init_IRQ()则是对中断控制器进行初始化,同时由于各种中断源共享 CPU 的中断请求引脚,所以需要为这些中断源建立一个中断请求队列,也就是一个结构数组 irq_list[]。这个数组的初始化在 init_IRQ()中完成。这部分的代码需要结合 S698 CPU 和其中断控制器的硬件结构加以修改,主要是做好对中断屏蔽和优先级、中断清除、中断挂起寄存器的编程控制。

在 SPARC 体系结构中,Trap 是进入超级用户模式的控制转移,其转移通过一个特殊 Trap 表来进行,Trap 由指令引起的异常或者外部中断请求引起,TYPE 值大于(或等于)0x80 的 Trap 是软 Trap[2]。需要根据处理器:

· 实现各种必要的 Trap 处理;

· 设置各个 Trap 矢量,矢量指向 Trap 处理句柄(如 2.2 节所述,这部分工作在引导代码中完成)。

要特别说明的是:S698 采用重叠寄存器窗口技术,当调用层数超过规定层时,会发生寄存器窗口溢出 Trap[2],这时需处理寄存器窗口溢出 Trap。处理的方法是,上溢时将先前寄存器窗口保存到堆栈中,下溢时将保存到堆栈中的原窗口数据取回恢复。

2.3.2 定时器

操作系统通过 time_init()函数对定时器初始化,其完成两个功能:1)初始化定时器,设定产生中断的周期。2)指定定时器的中断服务程序。在 init_IRQ() 中操作系统只是创建了一个中断请求队列,并没有指定某一个硬件中断具体对应于哪一个中断服务程序,所以必须在硬件的初始化程序或驱动程序中将具体的中断服务程序通过 reques_irq()向系统"登记",加入到中断请求队列当中。对于 S698 其实现如下:

```
request_irq(IRQ8, timer_interrupt, IRQ_FLG_LOCK, "timer", NULL);
```

然后初始化定时器相关寄存器(包括分时、计数、重装入、控制寄存器等)。

2.3.3 串行口

对控制台进行初始化的工作是在 console_init()函数中进行的。uClinux 默认的控制台输入输出终端都是串口,所以控制台初始化就是串口的初始化。类似于定时器的初始化,console_init()完成串口硬件初始化和登记中断服务程序这两个功能。串口初始化程序和串口中断服务程序都属于串口的驱动程序的一部分。根据 S698 中的 UART 添加串口驱动程序,并放在 Linux-2.0.x/drivers/char 下。

2.4 romfs 文件系统

uClinux 采用 romfs 文件系统,这种文件系统相对于一般的 ext2 文件系统要求更少的空间。romfs 文件系统不支持动态擦写保存,对于系统需要动态保存的数据采用虚拟 ram 盘的方法进行处理(ram 盘将采用 ext2 文件系统)。genromfs 把根文件系统生成 romfs.img。可以修改 Vendors 下的 Makefile 来设计自己需要的根文件系统。

2.5 操作系统映像的构建

uClinux 在构建内核时,首先建造内核的各个子系统部件,然后各个目标文件按照链接脚本文件(.ld)链接,形成最后可运行的程序。通过下面语句我们将上述的引导代码链接到最前面,其中 crt0_$(MODEL).o 是引导代码的目标文件。

HEAD := arch/$(ARCH)/platform/$(PLATFORM)/$(BOARD)/crt0_$(MODEL).o

链接脚本文件定义内存区(SECTION),段的运行地址(VMA)和装载地址(LMA),段的连接方式。一般来说,段的 VMA 和 LMA 可以一致,但对于可能改变内容的数据段、堆栈段,其 VMA 就必须设在 RAM 中某一地址。对于不同的目标板一般都需要修改链接脚本.ld 文件。这里,我们将.text 和.romvec装入在 Flash 的开始处(即零地址),VMA 和 LMA 一致,.data 数据段装入在 Flash 尾部,VMA 设在 RAM 某地址。

将 Trap vector,.text 和.data 段一起编译成为 kernel。用户程序可以被编译成 flat binary,放在 romfs.img 中,最后连接到原 Linux 的.text 段后面,成为最终的 image.bin。最终的 Flash 中各部件布置如图 2。

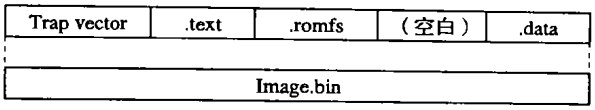

图 2　各部件布置图

3　结语

本文研究分析了 uClinux2.0.x 向 S698 移植中的几个主要问题,并给出了具体步骤。作者按此方法将 uClinux 2.0.x 内核成功移植至 S698 应用开发板上,并稳定运行。

参考文献:

[1] 欧比特(珠海)软件工程有限公司. S698 处理器应用开发系统用户手册[Z],2003.

[2] SPARC International Inc. The SPARC Architecture Manual Version 8[Z].

[3] Orbita Software Engineering INC. Sailing S698 User's Manual[Z], 2002.

[4] 毛德操,胡希明. Linux 内核源代码情景分析[M]. 第 3 版. 杭州:浙江大学出版社,2002.

(上接第 1035 页)

```
/* 给定目标物的初速度 */
a_mn[0] = dlg.m_xspeed;
a_mn[1] = dlg.m_yspeed;
a_mn[2] = dlg.m_zspeed;
if(a_mn[0]! =0) {
  /* 取目标物的初速度 */
  targetz = a_mn[0];
  targety = a_mn[1];
  targetx = a_mn[2];
  Mm mn;
  /* 取出计算结果 */
  dMm(m_sn);
  m_sn = Algol(a_mn[0], a_mn[1], a_mn[2]);
  ┆
  PostMessage(WM_PAINT);
}}
```

在类 CTwoHandsRobotView 的构造函数和析构函数中分别加入"initM(MATCOM_VERSION);"、"exitM();",用来初始化和结束调用"MATCOM"。

4　仿真结果

整个系统的动力学参数如下:

(1) 本体:$m_0 = 3.0\text{kg}; l_0 = 200$。

(2) 目标物:$m_c = 1.0\text{kg}; l_c = 100$。

(3) 右臂:$m_{11} = 1.0\text{kg}; l_{11} = 100; m_{12} = 1.0\text{kg}; l_{12} = 100; m_{13} = 0.5\text{kg}; l_{13} = 50$。

(4) 左臂:$m_{21} = 1.0\text{kg}; l_{21} = 100; m_{22} = 1.0\text{kg}; l_{22} = 100; m_{23} = 0.5\text{kg}; l_{23} = 50$。

初始条件:$r_0 = [0,0]^T, \phi_0 = 0, \phi_{11} = 45°, \phi_{12} = 90°, \phi_{13} = 45°, \phi_{21} = 135°, \phi_{22} = -90°, \phi_{23} = -45°$。

图 3 ~ 6 给出了 $V_c = [0.08, 0.08, 0.08]^T$ 时操作过程仿真结果。

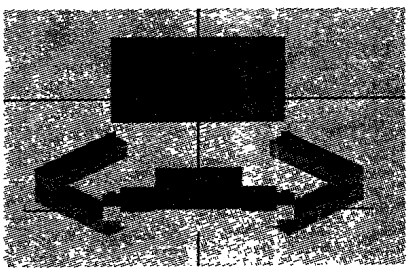

图 3　原始状态

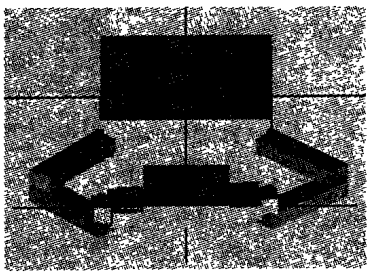

图 4　变化后的状态

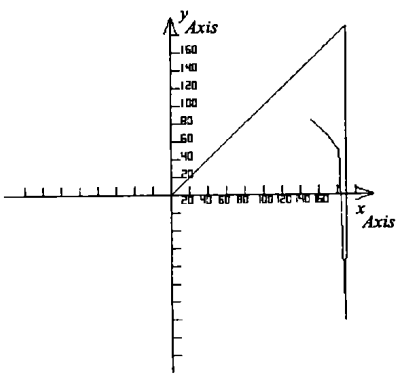

图 5　本体中心转角变化曲线

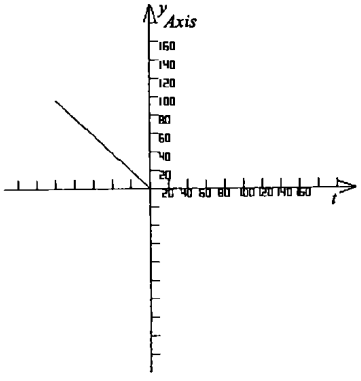

图 6　本体中心位姿变化曲线

5　结语

本文阐述了将 VC++ 6.0、Matlab 6.1、Matcom、OpenGL 融合到一起的方法和编程环境的配置,建立了一种研究机器人技术的仿真途径。在此平台基础上,实现了双臂机器人协调操作移动目标物的仿真运动。验证了该运动控制算法的正确性,以及通过这个方式编程效率等的优越性。

参考文献:

[1] 陈哲,吉熙章. 机器人技术基础[M]. 北京:机械工业出版社,1997.

[2] 郝红伟. MATLAB 6 实例教程[M]. 北京:中国电力出版社,2001.

[3] 张小,原义光,等. Matlab, Matcom 与 VC++ 接口的实现[J]. 计算机应用,2001,21(8).

[4] 白燕斌,史惠康,等. OpenGL 三维图形库编程指南[M]. 北京:机械工业出版社,1998.

[5] 黄维通,关继来. 边学边用 Visual C++ 编程[M]. 北京:清华大学出版社,2001.

[6] HU YAN-RU. George Vukovich. Dynamic Control Of Free - Flying Coordinated Space Robots[A]. Proceedings of the 34th Conference On Decision & Control New Orleans[C]. LA-December 1995.