S3C44B0X的μC/OS-II移植技术研究

李江乐,宗 容,裴以建,史晓敏
(云南大学 信息学院,云南 昆明 650091)

摘 要:在嵌入式系统的开发过程中由于操作系统的兼容性较差,经常会遇到基于一种CPU开发的操作系统不能在其他CPU上稳定的运行的情况,造成了重复开发的浪费。针对以上问题,对开放源码的μC/OS-II实时操作系统在三星公司的S3C44B0X处理器上的移植技术和移植条件进行了分析与研究,实现了μC/OS-II在S3C44B0X处理器上的移植,并对移植过程中的注意事项进行了讨论。实验证明,移植后的系统运行稳定和重新开发相比节省了大量时间,该移植方法对其他嵌入式操作系统的移植也有很好的借鉴作用。

关键词:μC/OS-II;S3C44B0X;实时操作系统;移植
中图分类号:TP316.2 **文献标识码**:A **文章编号**:1673-629X(2008)08-0134-03

Transplant Technique Research of μC/OS-II on S3C44B0X Microprocessor

LI Jiang-le,ZONG Rong,PEI Yi-jian,SHI Xiao-min
(College of Information, Yunnan University,Kunming 650091,China)

Abstract: In the development of an embedded system, the operating system developed on one type of CPU can not be manipulated well on another type and it causes the waste of time to re-develop a new system. To solve this problem, introduces and analyses the technique and necessary condition of transplanting the open-sourced embedded RTOS system μC/OS-II onto the S3C44B0X microprocessor. What should be paid attention to when transplanting is also discussed in the paper. The experimental result shows the realization of the transplantation and the system stability. Compared with re-developing a new system, this method saves a great amount of time and can be a useful reference to the transplantation of other embedded operating systems.

Key words: μC/OS-II;S3C44B0X; RTOS; transplant

0 引 言

随着嵌入式系统在各个领域的广泛应用,嵌入式操作系统的开发和移植成为了当今的研究热点。嵌入式操作系统是一种与系统硬件密切相关的软件,根据某种处理器设计的操作系统一般不能在其他种类的处理器上稳定运行。如果要在其他处理器上运行已经设计好的操作系统,就必须对它做相应的改造,这就是所谓的嵌入式操作系统的移植。文中重点针对在电子领域应用广泛的S3C44B0X处理器的μC/OS-II移植技术进行研究。

收稿日期:2007-11-30
基金项目:云南省自然科学基金资助项目(2004F0010M);云南大学重点资助项目(2003Z009B)
作者简介:李江乐(1978-),男,硕士研究生,主要从事自动控制理论及嵌入式系统的开发;宗 容,副教授,主要从事光纤通信及信号与信息处理理论的研究。

1 S3C44B0X和μC/OS-II介绍

S3C44B0X是SAMSUNG公司推出的一款16/32位RISC处理器,采用了ARM7TDMI内核,最大时钟频率可达75MHz,并提供了丰富的片上外设,如8k cache、LCD控制器、ADC、RTC、PWM、PLL等。通过提供全面通用的片上外设,大大减小了系统电路中除处理器以外的元器件配置,降低了系统成本[1]。

μC/OS-II是一个完整的,可移植、固化、裁减的先占式实时多任务内核。μC/OS-II是用ANSI的C语言编写,包含少部分的汇编代码,使之可在不同架构的处理器上使用。至今,μC/OS-II已经在从8位到64位机超过40种微处理器上移植成功。μC/OS-II的特点是:源代码公开、可移植性好、可固化、可裁减、多任务(最多可管理64个任务)、实时性强、稳定性好。

2 μC/OS-II的移植

下面对移植μC/OS-II的处理器硬件应具备的

条件和移植工作内容及移植技术,进行详细阐述。

2.1 μC/OS－II 的移植条件

大部分的 μC/OS－II 代码是用 C 语言编写的,但是与处理器相关的一些代码仍需要用汇编语言编写,这是因为操作系统在对 CPU 的寄存器进行直接操作时,必须用汇编语言来实现。同时,要使 μC/OS－II 能正常运行,被移植的处理器还要满足以下条件:

(1)处理器的 C 语言编译器能够产生可重入型代码;

(2)处理器能产生中断,并且能够产生定时中断;

(3)用 C 语言就可以开/关中断;

(4)处理器能支持一定数量的数据存储硬件堆栈;

(5)处理器有将堆栈指针以及其他 CPU 寄存器的内容读出并存储到堆栈或内存中去的指令。

本研究在移植过程中对不同公司和系列的处理器进行了对照研究,认为 SAMSUNG 公司的 S3C44B0X 处理器完全满足以上条件。而像 MOTOROLA 6805 系列的处理器就不能满足条件的后两项要求,所以不可以移植 μC/OS－II 操作系统在这类处理器上运行[2]。

2.2 μC/OS－II 的移植工作内容

本研究设计的整个 μC/OS－II 软硬件结构体系如图 1 所示。

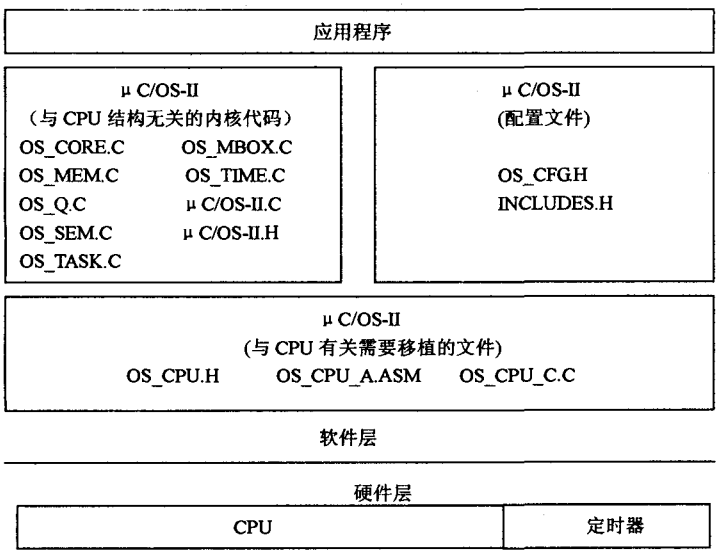

图 1　μC/OS－II 的软硬件体系结构图

如上图所示,移植工作主要是对 OS_CPU.H,OS_CPU_A.ASM,OS_CPU_C.C 三个文件的整改。具体工作内容包括:

用#define 设置 1 个常量的值;声明 10 个与 CPU 相关的数据类型;用 C 语言编写 6 个函数;编写 4 个汇编语言的函数[3]。

具体的移植工作的项目如表 1 所示。

表 1　具体移植工作项目说明

名称	类型	所在文件	语言种类	功能说明
BOOLEAN	数据类型	OS_CPU.H	C语言	
INT8U	数据类型	OS_CPU.H	C语言	
INT8S	数据类型	OS_CPU.H	C语言	
INT16U	数据类型	OS_CPU.H	C语言	
INT16S	数据类型	OS_CPU.H	C语言	
INT32U	数据类型	OS_CPU.H	C语言	
INT32S	数据类型	OS_CPU.H	C语言	
FP32	数据类型	OS_CPU.H	C语言	
FP64	数据类型	OS_CPU.H	C语言	
OS_STK	数据类型	OS_CPU.H	C语言	堆栈类型定义
OS_STK_GROWTH	宏定义	OS_CPU.H	C语言	堆栈增长方向
OS_ENTER_CRITICAL()	宏	OS_CPU.H	C语言	进入临界区
OS_EXIT_CRITICAL()	宏	OS_CPU.H	C语言	退出临界区
OSStartHighRdy()	函数	OS_CPU_A.ASM	汇编	启动任务
OSCtxSw()	函数	OS_CPU_A.ASM	汇编	任务切换
OSIntCtxSW()	函数	OS_CPU_A.ASM	汇编	中断退出时任务切换
OSTickISR()	函数	OS_CPU_A.ASM	汇编	系统时钟中断服务
OSTaskStkInit()	函数	OS_CPU_C.C	C语言	任务堆栈初始化
其他 5 个 HOOK 函数可省略	函数	OS_CPU_C.C	C语言	用户定义的钩子函数

2.3 μC/OS－II 的移植过程

μC/OS－II 的移植涉及到与编译器相关的数据类型的定义和处理器相关的汇编函数的定义,用于系统裁剪的配置文件和头文件的编写[4]。

2.3.1 相关的数据类型及宏定义

前 9 个数据是对与编译器相关数据类型的定义,本次移植采用的编译器是 SDT2.51,相关数据类型定义如下:

typedef unsigned char BOOLEAN; typedef unsigned char INT8U;

typedef signed char INT8S; typedef unsigned int INT16U;

typedef signed int INT16S; typedef unsigned long INT32U;

typedef signed long INT32S; typedef float FP32;

typedef double FP64;

OS_STK 是堆栈类型定义,它是堆栈的入口宽度与其 CPU 的类型有关,对于 S3C44B0X 应定义为 16 位宽:typedef unsigned int OS_STK。

常量 OS_STK－GROWTH 被定义位堆栈的增长方向,1 表示堆栈向下增长,0 表示向上增长。用宏定义实现:#define OS_STK－GROWTH 1。

OS－ENTER－CRITICAL()和 OS－EXIT－CRITICAL()宏的功能是实现程序进入临界区时对中断进行关和开的操作。定义如下:

#define OS－ENTER－CRITICAL() ARMDIsableInt()

#define OS-EXIT-CRITICAL() ARMEnableInt()

其中 ARMDIsableInt(),ARMEnableInt() 在 OS_CPU_A.ASM 中用汇编编写。

2.3.2 相关汇编函数的定义

对于 OSStartHighRdy(),OSCtxSw(),OSIntCtxSW()和 OSTickISR()这四个函数因为都涉及到对寄存器的操作,所以必须用汇编语言编写。

(1) 函数 OSStartHighRdy()在 μC/OS-II 运行过程中由 OSStar()函数调用,其功能是实现让系统运行优先级最高就绪状态的任务。

(2) 函数 OSCtxSw() 在 μC/OS-II 运行过程中通过执行软中断指令实现任务级的切换,中断服务子程序、陷阱、异常处理的向量地址必须指向 OSCtxSw()。

(3) 函数 OSIntCtxSW()在 μC/OS-II 运行过程中由 OSIntExit()函数调用,其功能是实现在 ISR(中断服务程序)执行任务切换功能。

(4) 函数 OSTickISR()为 μC/OS-II 提供一个周期性的时钟源,来实现时间的延迟和超时功能,时钟节拍必须在 OSStar()后启动[5]。

2.3.3 OSTaskStkInit()C 语言函数的定义

OSTaskCreate()函数通过调用 OSTaskStkInit()函数,初始化任务的堆栈结构。此时的堆栈看起来就同刚发生过中断一样,当前任务堆栈初始化完后,OSTaskStk Init 返回新的堆栈指针 STK,在 OSTaskCreate()执行时,将会调用 OSTaskStkInit()的初始化过程,然后通过调用 OSTCBInit()函数,将返回的 SP 指针保存到该任务的 TCB 模块中,初始状态的堆栈是模拟了一次中断后的堆栈结构[6]。

2.3.4 配置文件及头文件的编写

在实际的应用程序的设计中,一般不会用到 μC/OS-II 系统提供的全部函数,所以 μC/OS-II 允许用户根据实际需要对 μC/OS-II 进行裁剪,只选用应用程序用到的功能,而不用的功能通过条件编译命令裁剪掉[7]。这一功能是通过对配置文件 OS_CFG.H 的相关配置常量进行相应设置完成的,具体的定义是通过宏定义 #define 定义常量来实现的。这样就实现了 μC/OS-II 的可裁剪性。头文件主要是用来包含一些常用的库文件和用户自定义的子函数等,与平时用到的头文件差别不大。

经过上述的移植后,把移植好的文件和没有移植的与 CPU 无关的文件及用户应用程序加入到编译器进行编译连接就可以使 μC/OS-II 在 S3C44B0X 处理器系统上运行。

图 2 是移植在 S3C44B0X 上的 μC/OS-II 实现的多任务管理应用实例,通过 WINDOWS 超级终端利用串口通信观察到的运行结果。本实例由 5 个任务组成,其中一个主任务用来创建其他任务,一个任务扫描键盘,另外三个任务进行三路传感器模拟信号的 AD 转换[6]。

```
云南大学科学馆  616
-uC/OS-II test -
---Press  key3  to Start, Press  key2  to Stop---

***键盘扫描任务运行中 ! ***

 ***主任务运行中 ! ***

***任务一运行中 ! ***
 传感器一没有信号输出 !

***任务二运行中 ! ***
 传感器二没有信号输出 !

***任务三运行中 ! ***
 传感器三没有信号输出 !
```

图 2 μC/OS-II 多任务管理应用实例

3 在 μC/OS-II 移植过程中的注意事项

在 μC/OS-II 的移植和研究过程中,总结出以下 4 点 μC/OS-II 移植的注意事项:

(1)尽量减少任务的数量,相对少的任务数量可以减小任务间的切换次数,从而可以 减小系统的开销。

(2)合理的分配任务的优先级,一般把需要长时间运行的任务设置为比较低的优先级,可以减小其他任务的等待时间。

(3)掌握好开/关中断的时机,避免在中断服务函数中创建任务。

(4)原有的内核文件的组织缺乏条理性,在对系统进行移植过程中要注意在移植代码、内核代码、用户应用程序之间做好合理的划分和组织,这样有利于软件的维护和升级。

4 结束语

μC/OS-II 作为一个便于移植的多任务实时嵌入式操作系统,在中小型的工业领域应用广泛。文中所提到的移植方法较好地实现了 μC/OS-II 在 S3C44B0X 处理器上的移植。经过在试验开发板上的测试,证明该系统运行稳定,响应及时,达到了实时嵌入式操作系统的要求。另外,文中提出的移植注意事项是笔者在系统移植和开发实践中的经验总结,对其他处理器的 μC/OS-II 移植也有很好的参考价值。

(下转第 214 页)

便对服务提供者进行身份验证;

(5)EAAI 安全监控中心向服务提供者发送使用私钥进行加密的 SOAP 消息。

EAAI 安全方案还采用了另一种消息级别的安全形式——数字签名。数字签名就是附加给 SOAP 消息的证明真实性的数学语句。当接收系统获得消息和附加的数字签名时,它可以使用密钥来验证以下内容:请求者是消息的真正创建者(身份验证);SOAP 消息在传输过程中没有改变。数字签名和前面描述的完整加密过程之间的区别在于,如果使用数字签名,不必对整条消息进行加密。因此,系统的性能得到了提升。

3 应用案例与分析

3.1 EAAI 的应用案例

文中所述 EAAI 框架已经在中南大学相关教务系统的集成中得到了初步应用。原有的教务科管理的学籍管理系统、教务管理系统、选课系统均采用. Net 技术开发,而考试中心所属的大学英语四六级/计算机等级考试网上报名系统则采用 J2EE 平台开发,教务管理、选课及报名系统均需要使用学籍管理系统中学生基本信息,原来的解决办法是使用手工的方式拷贝数据,由于各系统中的学生信息会经常被修改,往往会产生严重的数据不一致问题,给使用和管理带来了麻烦。

通过引入面向服务的 EAAI 框架,首先对以上提及的四个系统进行了服务封装,在不影响已有应用的前提下,系统中的多个功能按照新的业务逻辑封装为 Web Service 以供调用,并且隐藏原有应用的复杂实现。开发了基于 J2EE 的业务服务总线及统一的教务门户网站,实现了服务共享和用户的单点登录。在安全方面也开发了安全监控软件,用一台独立的服务器来实现安全监控和服务代理的功能,同时使用 Apache 的 WSS4J 实现了基于 WS - Security 标准的消息级别的安全。

3.2 应用效果对比与分析

通过对基于 SOA 的 EAAI 在以上案例中的应用效果分析,参考文献[4]和文献[5],将 SOA 与传统组件技术的综合性能比较如表 1 所示。

表 1 与传统应用集成组件相比

比较项目	EJB	COM	CORBA	SOA
跨语言	一般	好	好	最好
跨平台	好	一般	好	最好
网络通讯	好	一般	最好	好
事物处理	一般	一般	好	一般
安全服务	好	一般	最好	好
可扩展性	一般	一般	好	最好
灵活性	一般	一般	好	最好
应用复杂度	高	低	高	最低

4 结束语

提出了一种新型的面向服务的电子教务应用集成框架 EAAI。EAAI 采用面向服务的体系结构,可以满足电子教务应用中对各种信息系统进行集成的要求,能动态适应教务流程的更新与快速响应,具有跨平台、灵活性、安全性、易扩展等特点。EAAI 框架在实践中的成功应用也证明了 EAAI 的优越性和有效性。下一步的工作是对 EAI 进行优化,提高 EAAI 的协作能力和事物处理能力,并增加对服务工作流的支持。

参考文献:

[1] 徐 罡,黄 涛,刘绍华,等. 分布应用集成核心技术研究综述[J]. 计算机学报, 2005,28(4):433 - 444.

[2] Perrey R, Lycett M. Service - oriented architecture[C]// Proceedings of the 2003 Symposium on Applications and the Internet Workshops. Orlando: IEEE Computer Society Press, 2003:116 - 119.

[3] Curbera F, Duftler M, Khalaf R. Unraveling the Web services Web: an introduction to SOAP, WSDL, and UDDI[J]. Internet Computing, IEEE,2002,6(2):86 - 93.

[4] 汪 芸,顾冠群. CORBA 技术综述[J]. 计算机科学,1999, 26(6):1 - 6.

[5] 童恒庆,聂会琴. CORBA/COM/EJB 三种组件模型的分析与比较[J]. 计算机应用研究,2004(4):66 - 67.

(上接第 136 页)

参考文献:

[1] 三星公司. S3C44B0X Data Sheet 数据手册[M]. 韩国:三星公司,2004.

[2] Labrosse J J. μC/OS - Ⅱ 嵌入式实时操作系统[M]. 邵贝贝译. 北京:北京航空航天大学出版社, 2005.

[3] 任 哲. 嵌入式操作系统 μC/OS - Ⅱ 原理及应用[M]. 北京:北京航空航天大学出版社,2005.

[4] Mike W, Barrett T. Embedded System Programming[J]. A Real - TimePrimer,1990,16(4):20 - 28.

[5] 黄燕平. μC/OS - Ⅱ ARM 移植要点详解[M]. 北京:北京航空航天大学出版社,2005.

[6] 胥 静. 嵌入式系统设计与开发实例详解[M]. 北京:北京航空航天大学出版社,2005.

[7] Brookshear J G. Computer Science: An Overview[M]. Sixth Edition. 北京:人民邮电出版社,2003.