广西科学院学报
Journal of Guangxi Academy of Sciences

2007,23(4):263~265
Vol. 23,No. 4 November 2007

μC/OS-Ⅱ在ARM7处理器上的移植
Transplanting μC/OS-Ⅱ to ARM7 Processor

赵晓鸥,黄福莹
ZHAO Xiao-ou,HUANG Fu-ying

(广西大学计算机与电子信息学院,广西南宁 530004)
(School of Computer, Electronics and Information, Guangxi University, Nanning, Guangxi, 530004, China)

摘要:以ARM7处理器为内核的LPC2104芯片为硬件平台,μC/OS-Ⅱ为软件平台,用ADS1.2编译器编写程序,设计并实现了μC/OS-Ⅱ操作系统到ARM7处理器的移植,并验证移植的正确性。将μC/OS-Ⅱ移植到ARM7处理器,可以充分发挥μC/OS-Ⅱ与ARM7的优势,有利于用户更有效地管理复杂的系统资源,使系统易于分层次处理,易于设计和维护。

关键词:μC/OS-Ⅱ ARM7 移植
中图法分类号:TP332 **文献标识码**:A **文章编号**:1002-7378(2007)04-0263-03

Abstract: The Transplantation μC/OS-Ⅱ to ARM7 Processor is designed, implemented by employing the LPC2104 with the core of processor as hardware platform, μC/OS-Ⅱ as software platform, ADS1.2 as procedures compiler. Transplanting μC/OS-Ⅱ to ARM7 processor can make best use of μC/OS-Ⅱ and ARM7 and enable the users to manage the system resources more effectively. The system is easy to handle at different levels, easy to design and maintain.
Key words: μC/OS-Ⅱ, ARM7, transplantation

嵌入式系统已在各个领域得到广泛应用。嵌入式实时操作系统的使用简化了嵌入式系统的应用开发,有效地确保系统的稳定性和可靠性,便于维护和二次开发。它的移植和扩展是近年来嵌入式系统领域的研究热点[1]。

ARM7是32位的微处理器,具有处理速度快、超低功耗、价格低廉和应用前景广泛等优点[2]。μC/OS-Ⅱ(微控制器操作系统版本2)是一个完整的、可移植、可固化、可剪裁的占先式实时多任务内核[3]。它公开所有的源代码,90%的代码使用标准的ANIS C语言书写,而且系统相对比较简单。将μC/OS-Ⅱ移植到ARM处理器,可以充分发挥两者的优势,有利于用户更有效地管理复杂的系统资源。

本文研究如何将μC/OS-Ⅱ移植到ARM7处理器,并通过编写应用程序以验证移植成功。移植的硬件平台采用高性能ARM7TDMI内核的LPC2104嵌入式处理器芯片,开发调试平台采用ARM ADS1.2,软件平台采用μC/OS-Ⅱ。

1 μC/OS-Ⅱ的体系结构

μC/OS-Ⅱ的软/硬件体系结构图参考文献[4]。μC/OS-Ⅱ的软件包括三部分代码:(1)μC/OS-Ⅱ核心代码,其中包括10个C程序文件和1个头文件,主要实现系统调度、任务管理、内存管理、信号量、消息邮箱和消息队列等系统功能;(2)μC/OS-Ⅱ配置代码,其中包括两个头文件,用于裁剪和配置μC/OS-Ⅱ,此部分代码与用户实际应用相关;(3)μC/OS-Ⅱ移植代码,其中包括1个汇编文件,1个C程序文件和1个头文件,这是移植μC/OS-Ⅱ所需要的代码,应根据目标处理器进行相应的编写、修改。

2 μC/OS-Ⅱ到ARM7的移植

将μC/OS-Ⅱ移植到新的体系结构上步骤是:首先,编写、修改μC/OS-Ⅱ源代码中的OS_CPU.H、OS_CPU_C.C、和OS_CPU_A.S三个文件;其

收稿日期:2007-07-30
作者简介:赵晓鸥(1984-),女,硕士研究生,主要从事嵌入式系统研究。

次，根据所使用的芯片获得相应启动代码；最后，根据μC/OS-Ⅱ的编程规范编写应用程序，验证移植是否成功。

2.1 OS_CPU.H的移植

该文件主要包含与编译器无关的数据类型的定义和与ARM7体系结构相关的定义。前者除了用typedef定义一些数据类型以外，还根据ARM7处理器的特性定义了堆栈的大小；后者定义了开/关中断的方式，并根据ADS1.2的特点定义堆栈的生长方向以及给各接口函数分配软中断功能号。

将μC/OS-Ⅱ移植到ARM7处理器上，首先必须在OS_CPU.H文件中完成以上定义。由于C语言中的short、int、long和char等数据类型依赖于编译器，与处理器的类型有关，即不可移植，所以在OS_CPU.H中定义了一系列不依赖于编译器、移植性强的数据类型为μC/OS-Ⅱ到ARM的移植服务。

另外，与所有实时内核一样，μC/OS-Ⅱ需要先关中断，再处理临界段代码，并且在处理完毕后重新开中断，使得临界代码免受多任务或中断服务子程序的破坏。但是由于各个编译器厂商使用的在C语言中执行开/关中断操作的实现方法不同，μC/OS-Ⅱ为了将其隐藏，增加可移植性，定义了用来开/关中断两个宏，即OS_ENTER_CRITICAL()和OS_EXIT_CRITICAL()。使用这两个宏的开/关中断的方式有三种，需要通过定义OS_CRITICAL_METHOD的值来确定。而常量OS_STK_GROWTH的定义确定了堆栈的生长方向，由于ADS1.2仅支持满堆栈递减，所以该常量赋值为1。

在移植过程中，为了使底层接口函数与处理器状态无关，同时在任务调用函数时不需要知道函数的位置，可使用软中断SWI作为底层接口，使用不同的功能号来实现函数的区分，同时预留挂接μC/OS-Ⅱ系统服务函数的接口。

2.2 OS_CPU_A.S的移植

该文件包括软中断SWI的汇编部分接口、函数OSStartHighRdy()的实现部分和中断退出时的任务切换函数OSIntCtxSw()三部分代码。

函数SoftwareInterrupt实现SWI的汇编部分接口，它判断任务的指令状态，然后取得对应的功能号，调用C语言部分代码实现系统服务的提供。

μC/OS-Ⅱ是抢占式多任务实时内核，得到CPU使用权并运行的始终是处于就绪状态下的优先级最高的任务。当某一正在运行的任务因某些原因脱离就绪状态，或者有一个较之优先级更高的任务进入就绪状态，那么此时就需要进行任务切换。任务切换有两种情况，一种是当前运行的任务主动放弃CPU控制权，这种情况通常发生在当前任务在等待某个事件或者调用系统延时的时候，此时调用函数OS_TASK_SW()实现任务切换；另一种是中断发生时，使得更高优先级的任务被激活进入就绪态，内核剥夺当前任务的CPU控制权使其失去运行权，即发生在中断退出时，此时调用函数OSIntCtxSw()实现任务切换。

OS_CPU_A.S文件的结构如图1表示。

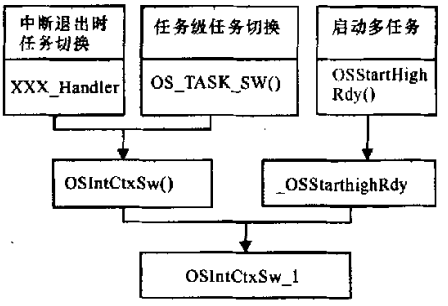

图1 文件OS_CPU_A.S的结构

2.3 OS_CPU_C.C的移植

μC/OS-Ⅱ的移植范例要求在编写OS_CPU_C.C文件时，需要编写10个函数，它们分别是任务堆栈初始化函数OSTaskStkInit()和另外9个μC/OS-Ⅱ在执行某些操作时调用的用户函数，即OSIniHookBegin()、OSInitHookEnd()、OSTaskCreat Hook()、OSTaskDelHook()、OSTaskSwHook()、OSTaskStatHook()、OSTCBInitHook()、OSTimeTickHook()和OSTaskIdleHook()。其中，函数OSTaskStkInit()是唯一必须保留的函数，其余9个函数必须声明，但是否包含这些函数代码由用户自己定。

移植使用SWI作为底层接口，而SWI软中断功能是在OS_CPU_C.C中实现的。应用函数SWI_Exception来实现软中断异常处理程序。

在μC/OS-Ⅱ源代码中已经声明函数OSStartHighRdy()，由于在C语言中用户不能直接访问CPU寄存器，所以还需在#.C文件中多加一次调用，使得这个函数的功能号在OS_CPU_A.S中实现。

2.4 获得LPC2104芯片的启动代码

启动代码是芯片复位后进入C语言程序的main()函数前执行的一段代码，主要为C语言程序提供基本运行环境。本文移植所使用的

EasyARM2104系统板附带有相关资料，包含有相应的LPC2100系列的启动代码，芯片的启动代码可以直接获得[5]。

3 应用程序的编写

使用的移植平台EasyARM2104的外设中，蜂鸣器、LED数码管和LED灯3个设备运行可以比较明显地显示出来，适合观察μC/OS-Ⅱ的多任务切换。编写的应用程序中包含的3个任务，即LED数码管显示0-F字符、LED灯以跑马灯形式显示0-F数据和蜂鸣器鸣叫。

用μC/OS-Ⅱ中的OSTaskSuspend()挂起任务函数和OSTaskResume()唤醒任务函数来实现多任务的嵌套。应用程序的编写过程：

首先，任务嵌套要实现的目标效果是：LED数码管显示——蜂鸣器鸣叫两声——LED灯显示——蜂鸣器鸣叫两声——LED数码管显示，…，这样往复循环切换。

其次，通过任务嵌套来完成目标效果的实现，实现步骤(图2)如下。

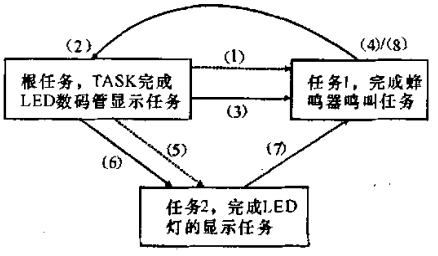

图2 多任务嵌套过程

(1)在主函数中创建根任务，这里将LED数码管显示任务作为根任务，并在进入该任务函数的死循环体去执行LED数码管显示函数前创建任务1(实现蜂鸣器鸣叫两声的任务)；

(2)在执行完(1)时由于任务1的优先级别低于根任务，所以系统继续往下执行根任务的程序代码，完成LED数码管显示；

(3)在执行完LED数码管显示函数后，使用挂起函数OSTaskSuspend()将根任务挂起，μC/OS-Ⅱ将CPU使用权交予任务1去执行蜂鸣器鸣叫函数；

(4)在任务1执行完蜂鸣器鸣叫后，用函数OSTaskResume()唤醒根任务，则CPU使用权返还根任务，接着执行步骤(3)中的挂起函数后面的程序代码；

(5)创建任务2，它主要实现LED灯的显示任务，其优先级高于任务1但低于根任务；

(6)完成任务2的创建后，根任务再次调用函数OSTaskSuspend()将自己挂起，而处于就绪状态的两个任务中由优先级高的任务2抢得CPU使用权，执行LED灯显示任务函数；

(7)任务2中的LED灯显示函数执行完后，任务通过删除自己实现μC/OS-Ⅱ规定的任务无返回值，同时向任务1出让CPU使用权；

(8)任务1执行完蜂鸣器鸣叫函数后，又再次唤醒根任务，这时候根任务返回到死循环体的开端重新进行相似的第二轮循环(该步骤路线和路线(4)所示实线相同)。

这样就实现了此次多任务的循环嵌套，任务间的切换由操作系统自行完成。

4 移植成果的验证

移植代码编写、修改完毕后，将所有代码集合到在ADS1.2中创建的工程里，通过JTAG仿真器全部烧写到实验板的Flash中(因为LPC2104芯片的内存容量较小，不够用)，脱离PC机运行，实验板上显示出预计效果。这样就验证移植已经成功完成，应用程序编写是正确的。

5 结束语

本文设计了μC/OS-Ⅱ操作系统到ARM7处理器的移植，以及移植结果在LPC2104板上的应用的方法和步骤。通过移植处理，使得用户更方便，更为有效地管理越来越复杂的系统资源。μC/OS-Ⅱ的多任务特点可以帮助我们简化复杂应用程序的编写，使系统易于分层次处理，易于设计和维护。

参考文献：

[1] 任哲. 嵌入式实时操作系统μC/OS-Ⅱ原理及应用[M]. 北京：北京航空航天大学出版社，2004.

[2] 周立功. ARM嵌入式系统基础教程[M]. 北京：北京航空航天大学出版社，2005.

[3] 周立功. ARM微控制器基础与实战[M]. 北京：北京航空航天大学出版社，2003.

[4] JEAN J LABROSSE. 嵌入式实时操作系统μC/OS-Ⅱ[M]. 邵贝贝，译. 第2版. 北京：北京航空航天大学出版社，2003.

[5] JEAN J LABROSSE. MicroC/OS-Ⅱ thereal-time kernel[M]. Second Edition. USA：Published by CMP Books，CMP Media LLC，2002.

(责任编辑：尹 闯)