文章编号: 1007-9432(2006) 01-0100-04

嵌入式操作系统 μC/OS-Ⅱ在单片机上的移植

靳文兵, 白凤娥, 潘志栋
(太原理工大学 计算机与软件学院, 山西 太原 030024)

摘 要: 简要介绍μC/OS-Ⅱ的组成和工作原理, 详细分析移植的环境和条件, 明确指出移植中的关键问题, 即堆栈的建立、释放和恢复问题。以 51 单片机系统为例, 提出了相应的解决办法, 并对编制的程序代码进行了说明。通过在 8031 单片机系统中建立工作堆栈、仿真堆栈、任务堆栈及其附加段, 修改μC/OS-Ⅱ中与移植有关的程序文件, 实现了μC/OS-Ⅱ在单片机系统中的移植。相关实验测试进一步证实了本移植思想。文章深入讨论部分为嵌入式操作系统μC/OS-Ⅱ的复杂应用提供了指导。

关键词: 单片机; 嵌入式系统; 操作系统; 移植; μC/OS-Ⅱ
中图分类号: TP368; TP311.54 文献标识码: A

嵌入式实时操作系统在许多领域得到广泛应用。μC/OS-Ⅱ是一种源代码公开的可移植、可裁剪的占先式多任务嵌入式实时操作系统[1], 其功能完备、性能可靠、短小精炼, 与其它嵌入式实时操作系统相比, 它更适用于硬件资源有限的单片机系统。

在国内, 随着高速、高性能 51 系列单片机的出现, 使 51 系列单片机的应用更趋广泛。为 51 单片机系统移植嵌入式实时操作系统, 具有一定的现实意义。由于单片机硬件系统提供的可利用资源相对较少[2], 将μC/OS-Ⅱ移植到 51 单片机系统中, 其移植方法有许多特别之处。

1 μC/OS-Ⅱ工作原理

1.1 μC/OS-Ⅱ的组成

按功能分类, μC/OS-Ⅱ主要由 5 部分组成:

1) 核心部分(OS_CORE.C): 完成操作系统初始化、任务调度等工作。

2) 任务部分(OS_TASK.C): 完成任务建立、删除、挂起和恢复等工作。

3) 时钟部分(OS_TIME.C): 完成任务延时和时间控制等工作。

4) 任务通信和内存管理部分(OS_FLAG.C、OS_MBOX.C、OS_MEM.C、OS_SEM.C、OS_Q.C、OS_MUTEX.C): 完成协调任务和临界资源管理等工作。

5) 与移植相关部分(OS_CPU_C.C、OS_CPU_A.ASM): 需要根据不同的 CPU 来编制实现堆栈建立的函数和底层任务切换、时钟节拍中断函数等; 另外还需要根据应用不同而更改或调整部分头函数(OS_CFG.H、UCOS_II.H、OS_CPU.H)的内容。

1.2 μC/OS-Ⅱ的任务调度机制

调度是指任务间的转换, 即保存一个任务的执行环境并恢复另一个任务的执行环境, 也称任务上下文的切换, 其本质是任务堆栈与工作堆栈的相互复制。任务堆栈是每一个任务应有的、用于保存其执行时的上下文环境参数, 包括其被中断的地址、工作寄存器和局部变量的值等信息。任务堆栈的数量与任务数量相等, 系统任务也要有它们各自的任务堆栈。工作堆栈则是正在占用 CPU 的任务所使用的堆栈, 也称硬件堆栈或系统堆栈。每台机器或者说每个系统只有一个工作堆栈, CPU 通过堆栈指针操作工作堆栈的内容, 如图 1 中(1)、(2)两部分所示。箭头组①是指工作堆栈的内容向任务堆栈的复制, 而②则恰恰相反。每一次任务的调度都要进行任务上下文的保存和切换工作, 即将当前工作堆栈的内容复制到旧任务的任务堆栈中, 同时也把新任务的任务堆栈内容装入工作堆栈中去。

μC/OS-Ⅱ操作系统总是调度当前就绪队列中优先级最高的任务, 并使其作为新任务进入执行状态并开始占用 CPU 资源。调度发生在每一个时钟

收稿日期: 2005-03-22
作者简介: 靳文兵(1968-), 男, 山西洪洞人, 硕士, 主要从事计算机监控研究, (Tel) 13603518440, (E-mail) wenbbo@sina.com
通讯联系人: 白凤娥, 教授, (Tel) 13835102363, (E-mail) baifl@sina.com

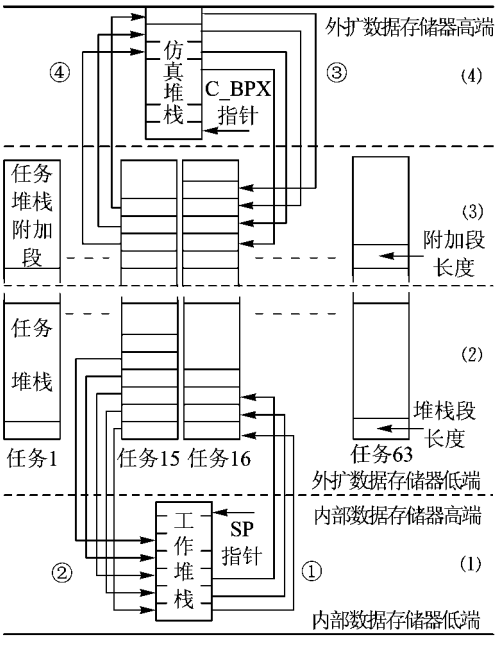

图 1　各堆栈之间的关系

节拍中断中, 或在任务有函数调用后。这时, 就绪队列中优先级均有可能发生变化。在其它情况下, 也可能发生就绪队列优先级的变化, 如当前执行任务对临界资源释放时引起的变化。用时钟节拍中断来实现调度, 使系统对任务的实时调度响应时间大约在一个时钟节拍左右, 以满足系统实时性的要求。μC/ OS-Ⅱ中任务的优先级编号是唯一的, 而任务编号和优先级编号可以不同。

1.3　μC/ OS-Ⅱ的移植条件

对 μC/ OS-Ⅱ进行移植时, 必须满足 5 个条件:

1) 处理器 C 编译器支持可重入代码的生成。系统功能函数或任务程序应具有可重入性, 这是实时系统及时发生中断调度所需要的, 也是可剥夺型内核的特点。这样, 任务程序或系统功能函数可被中断, 中断的任务程序或系统功能函数的局部变量保存在适当的地方, 下次任务程序被恢复时, 取出这些保存的局部变量, 任务就可继续执行。

2) 用 C 语言可以打开和关闭中断。

3) 处理器支持中断, 并能产生定时中断, 中断频率通常设置在 10~ 100 Hz 之间。

4) 处理器支持足够的 RAM 空间, 以满足多任务环境下设置任务堆栈的要求。

5) 处理器有相应的指令, 能将堆栈指针和其它 CPU 寄存器读出和存储到堆栈或内存中。

2　μC/ OS-Ⅱ的移植环境

2.1　硬件资源

通过 PC 将 μC/ OS-Ⅱ移植到 51 单片机系统中, 该系统主要包括 4 部分:

1) Intel 8031 处理器, 内含 128 字节 RAM 存储器, 是单片机系统的控制核心。

2) 8031 外部扩展数据/ 程序存储器。程序段地址 0H~ 4000H 为实验板系统占用, 4000H~ 0BFFFH, 提供实验用程序段和数据段。

3) 两片 74LS274 芯片, 分别可编址, 供 8031 扩展输出口。

4) 发光二极管指示灯 12 只, 可展示三个任务的工作状态。

2.2　软件资源

1) 嵌入式实时操作系统 μC/ OS-Ⅱ V2.52 源代码。移植主要利用其核心部分, 即任务调度和部分时间管理代码。

2) Keil Elektronik GmbH μVision2 集成编译环境, 完成源程序的交叉编译工作。

3) TMSD. EXE 程序, 完成目标程序由 PC 下载到单片机程序存储器的工作。

3　移植中的关键问题

3.1　对移植条件的分析

51 单片机系统的编译软件如 μVision2 通过构造仿真堆栈来解决函数的可重入问题。其专用 C 语言, 对中断开关操作的实现, 与 ANSIC 基本相同。51 单片机都支持中断, 包括定时中断, 并可支持外扩 64 kB 的最大数据存储器空间, 满足 μC/ OS-Ⅱ系统的移植条件。

3.2　工作堆栈和仿真堆栈的建立

工作堆栈的框架由 51 单片机决定, 只能建立在单片机内部数据存储器中, CPU 通过堆栈指针可以读写工作堆栈的内容。因为工作堆栈容量最大 128 字节, 为了保证工作堆栈有足够大的空间, 避免在多次程序嵌套时产生溢出, 可通过一个全局变量来限制程序嵌套的层数。

51 单片机利用仿真堆栈存储可重入函数的局部变量。仿真堆栈不同于工作堆栈, 它的框架实际由编译系统软件如 μVision2 提供。因单片机内部数据存储器容量的限制, 一般把仿真堆栈建立在单片机外部数据存储器的高端, 由上向下生长。CPU 可以通过 C_XBP 指针读写仿真堆栈的内容。

3.3　任务堆栈和任务堆栈附加段的建立

51 单片机内部 128 字节的数据存储器仅适合安排工作堆栈和存放部分全局变量, 所以任务堆栈应建立在外部数据存储器中。任务堆栈一般是以静

态数组的形式建立起来,既便于初始化和定位,又便于控制其深度。每一个任务均有相应任务堆栈指针指向其任务堆栈的初始位置。任务堆栈指针被保存在任务控制块中,每一个任务控制块管理一个任务。任务控制块也建立在外部数据存储器中。

任务堆栈附加段可以放在每一个任务堆栈的尾部,它是用来保存任务被中断时其仿真堆栈中的内容。图1中(3)、(4)两部分反映了任务堆栈附加段与仿真堆栈间的保存和恢复工作,其中箭头组③指仿真堆栈内容向任务堆栈附加段的复制,而④则指任务堆栈附加段内容向仿真堆栈的复制。

对于51单片机系统,每一次任务调度时的上下文切换工作,既包括任务堆栈和工作堆栈的内容切换,也包括任务堆栈附加段和仿真堆栈内容的切换。

3.4 任务调度的机理

当系统发生任务调度时,旧任务首先把其部分上下文环境参数压入工作堆栈中,这些参数包括任务程序中断的返回地址,也就是下一次本任务被调度后的起始工作地址,还有所有寄存器内容。另外还有一部分上下文环境参数涉及局部变量,它们是保存在仿真堆栈中的,并会在任务切换时,被复制到相应的旧任务的任务堆栈附加段中。全部工作堆栈的内容将被复制到外部数据存储器中旧任务的任务堆栈中。然后,新任务把其存放于外部数据存储器中的任务堆栈内容复制到内部工作堆栈中。通过适当调整堆栈指针值,可用中断返回指令来正确建立部分上下文环境和恢复任务程序在上次调度时被中断了的执行地址。而其他上下文环境参数则通过恢复仿真堆栈指针和把任务堆栈的附加段复制到仿真堆栈中去来实现。

4 移植代码说明

μC/OS-Ⅱ移植过程中需要修改的文件有五个,即OS_CPU_C.C,OS_CPU_A.ASM,OS_CPU.H、OS_CFG.H和UCOS_II.H。

4.1 OS_CPU_C.C文件

移植μC/OS-Ⅱ时,在OS_CPU_C.C文件需要定义六个函数,实际只需定义OSTaskStkInit()函数,它被OSTaskCreat()函数或OSTaskCreatExt()函数调用。为任务建立任务堆栈,模仿已发生过中断后的入栈顺序[1],栈长可通过指向静态数组下标来调整。每一个静态数组被分成两部分,前半部分属于任务堆栈,后半部分属于任务堆栈附加段。也即只需通过预留空间即可建立任务堆栈附加段。任务堆栈和任务堆栈附加段可用一个指针来访问。

4.2 OS_CPU_A.ASM文件

OS_CPU_A.ASM文件主要包括5类函数。

1) 初始任务调度函数OSStartHighRdy()。OSStartHighRdy()函数被OSStart()函数调用,在程序开始时启动优先级最高的任务。

2) 任务级任务切换函数OSCtxSw()。OSCtxSw()函数用于替换宏OS_TASK_SW()。它在需要任务级任务切换时被调用,以切换任务上下文环境,如有函数调用而发生优先级的转化时。也即完成工作堆栈与任务堆栈的切换和仿真堆栈与任务堆栈附加段的切换。

3) 中断级任务切换函数OSCtxIntSw()。OSCtxIntSw()函数与OSCtxSw()函数的构成和作用相似,是中断级任务切换函数,并在中断发生后期被OSIntExit()函数调用,以完成中断级切换任务上下文环境。

4) 时钟节拍中断函数OSTickISR()。OSTickISR()函数定位于51单片机的定时器中断1上,它通过调用OSTimeTick()函数,为操作系统提供周期性的时钟源,以实现任务时间的延迟和超时功能。

4.3 OS_CPU.H、OS_CFG.H和UCOS_II.H文件

OS_CPU.H中定义了与处理器相关的常数、宏以及类型,需根据51单片机作出适当调整;OS_CFG.H中多是系统可裁剪部分的开关量值,可根据需要调整;在UCOS_II.H中可定义结构体,将变量放在内部或外部扩展数据存储器中。因51单片机内部数据存储器空间有限,故将绝大部分结构体和变量都放在外部扩展数据存储器中。

4.4 板级支持函数port1()、port11()、port2()、port21()

这类函数为任务提供移植目标系统的板级接口,通过地址1B08H和1B10H外接发光二极管,用来显示各任务执行情况。它们是为显示移植结果而自行编制的程序。

5 移植结果测试

5.1 任务调度的测试

基本任务堆栈到工作堆栈的测试,使用系统自测的方法进行[1]。

5.2 仿真堆栈的测试

可重入函数或任务的局部变量是否可以正常保存和恢复是移植的重点。为此,建立两个任务Task1()和Task2(),它们均有局部变量。对单一仿

真堆栈的利用, 两者将出现相互覆盖对方数据的现象。设计 Task1() 通过处理器 8031 的 P1 口直接连接 8 只发光二极管, 用局部变量 i 控制循环语句, 实现发光二极管逐位顺序点亮。局部变量保存和恢复不正确将使发光二极管乱序点亮。操作系统调度 Task2( ) 和 OSTaskIdle( ) 将使由地址 1B08H 和 1B10H 扩展的板级接口输出高、低电平, 它们被分别连接到四只发光二极管上, 因此二极管呈现出不同频率的闪烁状态。经测试, 下载到实验系统上的移植程序可以顺序点亮 P1 口的 8 只发光二极管; 另外四只发光二极管分成两组, 以不同的频率闪烁。实验时间延续长达 50 小时, 无乱频闪烁现象。实验证明移植成功。

6　深入讨论

6.1　堆栈长度的确定

堆栈总是越长越好, 但现实是不允许的。μC/ OS-Ⅱ提供了一个函数 OSTaskStkTest( ), 专门用于测试所需任务堆栈的长度, 只需稍加改动, 它一样可以被用来测试任务堆栈附加段的长度。在堆栈长度不确定时, 堆栈及其附加段空间应设置尽量大些。如果为附加段提供一个指针, 可以提高系统查询效率; 如果把附加段和仿真堆栈合并, 那样将更好。

6.2　函数嵌套的层数

在任务调度过程中, 必须限制函数嵌套的层数, 否则系统很容易因堆栈溢出而崩溃。工作堆栈一般不会超过 80 字节, 如果每层 15 字节, 嵌套不应超过 5 层。可以设计一个全局变量来控制嵌套的层数。

6.3　中断开关的时机

在多任务的调度过程中, 如果对中断的开或关处理不当将会造成系统死机, 为避免这种情况的发生, 往往要采取一些相应的措施, 如不在任务建立前打开定时器开关, 以免造成提前对任务进行调度。这个开关不是指 EA, 而是指 ET, 用 TR 也可以, 因为在第一个任务建立前, EA 已经被打开过了。

6.4　存储覆盖的问题

在 μC/ OS-Ⅱ移植到单片机时, 当仿真堆栈向下生长时, 可能与存储器中的其它变量相互覆盖, 还有可能造成其它数据结构和变量之间的覆盖。对编译系统来说, 解决这一问题的最好办法是把部分失控的仿真堆栈安排在存储器的顶部, 另外, 利用操作系统提供的存储器管理功能, 也可以解决这方面的问题。

7　结语

通过在单片机系统的数据存储器中建立工作堆栈、仿真堆栈、任务堆栈及其附加段, 实现了 μC/ OS-Ⅱ嵌入式实时操作系统向 51 系列单片机的移植, 从而为 51 系列单片机的复杂应用, 提供了基本的条件。

参考文献:

[1]　Jean J Labrosse, 邵贝贝译. 嵌入式实时操作系统μC/ OS-Ⅱ[M]. 北京: 北京航空航天大学出版社, 2004.

[2]　薛钧义. 微机控制系统及其应用[M]. 西安: 西安交通大学出版社, 2003.

Embedded Operating System μC/ OS-Ⅱ Porting to SCM

JIN Wen-bing, BAI Feng-e, PAN Zhi-dong

(College of Computer and Software, Taiyuan University of Technology, Taiyuan 030024, China)

Abstract: This paper introduces μC/ OS-Ⅱ and how to port it to Intel 51-serie single-chip system by means of establishing working stack、simμlation stack、task stack and its additional part in the single-chip system. Analysis and interpretation offered in detail for the relative programmed code as well as this porting method. Further discussions at the end of this paper refers to complicated applications.

Key words: single-chip; embedded system; operating system; porting; μC/ OS-Ⅱ

(编辑: 刘笑达)