

μC/OS-II 应用于 8051 单片机的优缺点

柏铁山

(常州轻工职业技术学院 信息工程系 江苏 常州 213164)

摘要 当前 8051 系列单片机的应用非常广泛, μC/OS-II 这一抢占式的嵌入式实时操作系统内核在实践应用中被证明是十分稳定可靠的一种系统内核, 将这两者相结合是很多 8051 系列单片机构架嵌入式系统的方式, 本文介绍了 μC/OS-II 具体应用于 8051 系列单片机的情况, 总结了一下两者结合的优点和缺点, 对于一些固有的缺陷也给出了具体的解决方案。

关键词 8051 单片机 嵌入式系统 μC/OS-II 移植

中图分类号: TP399

文献标识码: B

文章编号: 1672-6251(2009)03-0087-03

μC/OS-II 是一种源代码开放的实时操作系统, 支持多任务运行, 能够适应苛刻的实时性要求, 8051 是一种应用非常广泛的单片机, 将 μC/OS-II 应用于 8051 系列单片机构架的嵌入式系统是一种有益的尝试, 在实践应用过程中, 这种结合体现出了一定的优点, 但同时也存在一些固有缺陷, 下面就这些优缺点作一些总结。

1 μC/OS-II 在 8051 单片机上的移植

μC/OS-II 的移植是要满足一定要求的。

1.1 处理器的 C 编译器能产生可重入的代码

C51 的编译器是可以产生可重入代码的, 在可重入函数定义时加上 reentrant 关键词, 只是, 由于物理堆栈比较小, 对函数重入是通过在内存中开辟模拟堆栈来实现的, 因此, 效率比较低。

1.2 处理器支持中断, 并且能够产生定时中断

8051 系列单片机都是支持中断的, 一般的 8051 单片机都会提供一到两个定时/计数器, 并且, 这些定时/计数器都能提供定时中断, 通常典型的用法是, 一个作为定时器使用, 一个用作波特率发生器。

1.3 用 C 语言就可以开关中断

8051 系列单片机上, C 语言是可以直接操作单片机内部寄存器的, 开关中断正是通过操作相应的单片机内部寄存器来实现的, 因此, 在 8051 单片机上是可以直接用 C 语言直接开关中断的。

1.4 处理器能处理一定数量的数据存储硬件堆栈

8051 单片机是有硬件堆栈的, 但是, 容量非常小, 而函数重入和中断嵌套是需要堆栈支持的, 函数的多次重入和中断的多次嵌套需要容量较大的堆栈来保护现场, 比如保存函数的形参和局部变量。对于堆

栈容量过小, 是可以解决的, 可以在内部和外部存储区开辟开辟一个模拟堆栈, 只是, 这样做会带来系统的额外开销。

1.5 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

对于这一点, 8051 系列的单片机是可以满足的, 对于 CPU 中的部分寄存器甚至可以直接用 C 语言进行读写操作。

从上面的叙述可以看出, 在 8051 系列单片机上移植 μC/OS-II 是可行的, 具体的移植工作针对不同的处理器是不同的, 但大体来说有这样一些移植的操作。

数据的定义。要把 μC/OS-II 中使用的数据类型和具体处理器的数据长度统一起来, 同时要避免 μC/OS-II 中使用的一些变量、常量名称和编译器保留的关键词相冲突。

堆栈的设计。对于 μC/OS-II 的移植, 堆栈的处理始终是一个重点, 通常需要考虑不同处理器的系统环境, 使用 μC/OS-II, 往往是希望能在系统中体现 μC/OS-II 的实时性和多任务性, 多任务的切换就要考虑保存任务的当前运行环境。对于 8051 系列单片机, 系统环境通常体现于处理器中的各种寄存器, 此外, 考虑到 8051 系列单片机中的 C 语言函数重入问题, 系统环境还包括了模拟堆栈的内容。在设计堆栈时, 也要注意 8051 的硬件堆栈生长方向是从低地址向高地址扩展的, 对于 KEIL 的编译器, 模拟堆栈的生长方向则是从高地址向低地址扩展。

多 bank 的处理。对于有些高性能的 8051 单片机, 内部 flash 容量相对较大, 会在单片机内有多

收稿日期 2008-12-24 修回日期 2008-12-25

作者简介 柏铁山(1977-) 男, 讲师, 研究方向: 计算机应用。

bank 的结构, 在使用了 $\mu\text{C}/\text{OS-II}$ 后, 在发生任务切换时, 如果切换后执行的程序不在当前 bank 内, 就必然带来 bank 切换的问题, 这需要在移植时有特殊的处理, 这里, 有两个需要处理的细节: 第一, 在堆栈中保存环境时要包括和 bank 切换相关的寄存器; 第二, 在任务堆栈初始化时, 要求堆栈中保存有任务首条指令的运行环境, 这当然也包括了 bank 有关的属性, 由于初始化函数在运行时, 去取得保存 bank 属性的寄存器时, 此时寄存器中的内容是初始化函数所在 bank 的属性, 因此, 如果要保证任务首条指令的 bank 环境正确, 就要把首条指令所在函数和堆栈初始化函数在同一 bank 中, 或者, 至少要保证初始化函数能得到任务首条指令所在 bank 的属性, 以便能在初始化时保存。

其他细节的处理。在移植时, 如果在汇编语言中用到某些 $\mu\text{C}/\text{OS-II}$ 中的变量, 最好将其用 IDATA 定义, 可以让编程更加方便。

2 $\mu\text{C}/\text{OS-II}$ 与 8051 系列单片机结合的优点

8051 系列的单片机应用非常广泛, 在 8051 的家族里也不乏高性能、大容量的芯片, 这些大容量的芯片可以装入更大的程序, 实现更复杂的功能。在实际应用中随着代码量的增加, 就要求代码有更优良的组织结构, 并且对代码的可维护性也有了更高的要求。这样, 在 51 单片机上应用操作系统就变得切实可行。

对于 8051 系列单片机采用操作系统有很多的选择, 在选择这些操作系统时可以考虑很多因素, 主要来说可以考虑: 系统扩展性是否良好; 系统可维护性是否良好; 系统多任务处理是否完善; 系统实时性能是否良好等等。 $\mu\text{C}/\text{OS-II}$ 应用在 8051 系列单片机上是有优势的, 具体来说, 有以下一些优点:

2.1 $\mu\text{C}/\text{OS-II}$ 开放源代码

$\mu\text{C}/\text{OS-II}$ 不是免费的, 但性价比很高, 作为一种操作系统, 开放源代码是他的一个重要特点, 这在嵌入式系统中的应用是有实际意义的, 代码的开放, 意味着可以在开发过程中可以对代码进行跟踪调试, 可以对其代码进行有效的修改以适应系统运行, 程序员也可以最大限度地掌控操作系统运行。

2.2 $\mu\text{C}/\text{OS-II}$ 代码简洁高效

$\mu\text{C}/\text{OS-II}$ 的代码十分简洁, 代码量很小, 这就决定了, 在使用 $\mu\text{C}/\text{OS-II}$ 时, 可以快速地掌握这种操作系统的运行原理和性能, 能针对实际需求编写出高效的系统任务。而且, $\mu\text{C}/\text{OS-II}$ 编译后体积小, 不会

对系统造成太大的代码负担。

2.3 $\mu\text{C}/\text{OS-II}$ 是一种多任务操作系统

在嵌入式系统中决定是否使用操作系统, 往往是由于多任务的需求, 当然, 多任务性并不是只有操作系统可以提供, 但没有操作系统良好的组织, 多任务的实现往往造成代码的繁杂, 系统的可靠性自然也就降低。大多数的嵌入式操作系统都支持多任务。 $\mu\text{C}/\text{OS-II}$ 相比其他操作系统, 其多任务的实现简单可靠, 由于开放源代码, 其多任务的机制对程序员是透明的, 因此可维护性更好。在 $\mu\text{C}/\text{OS-II}$ 中, 任务的增加是非常简单的, 因此, 应用 $\mu\text{C}/\text{OS-II}$ 的嵌入式系统功能扩展性非常好。

2.4 $\mu\text{C}/\text{OS-II}$ 是一种实时操作系统

$\mu\text{C}/\text{OS-II}$ 实际上是一种可剥夺的实时操作系统, 系统在设计时就精确考虑了各种系统开销时间, 加上采用任务优先级唯一的机制, 其实时性是非常好的。8051 系列的单片机经常被用在各种实时嵌入式系统中, 采用 $\mu\text{C}/\text{OS-II}$ 可以在可控范围内较好地保证系统苛刻的实时性要求。

2.5 $\mu\text{C}/\text{OS-II}$ 是一种可靠而稳定的系统

$\mu\text{C}/\text{OS-II}$ 是一种应用非常广泛的操作系统, 在很多嵌入式设备中都有应用, 其中有些设备甚至在苛刻的环境中应用, 在长时间应用过程中, 证明了 $\mu\text{C}/\text{OS-II}$ 是一种可靠、高效、强壮的系统, 既然这种系统如此可靠, 当然也希望能将其应用到广泛使用的 8051 单片机上。

$\mu\text{C}/\text{OS-II}$ 应用在 8051 单片机上充分体现了 $\mu\text{C}/\text{OS-II}$ 的优点, 实践也证明了, 应用 $\mu\text{C}/\text{OS-II}$ 的 51 单片机系统能较为高效并且长时间稳定地运行, 对于有实时性要求的系统, 也能充分发挥 $\mu\text{C}/\text{OS-II}$ 的实时性能。

3 $\mu\text{C}/\text{OS-II}$ 在 8051 系列单片机上使用的缺陷

$\mu\text{C}/\text{OS-II}$ 的确是一种高效的实时嵌入式操作系统, 其优点, 在上文已有论述, 但其在 8051 系列单片机上移植应用时, 有着一些固有的缺陷, 这是由 8051 系列单片机的体系结构决定的, 在实际应用中也都有所体现, 对整个系统的构建有着很多的制约。

具体来说, $\mu\text{C}/\text{OS-II}$ 应用在 8051 单片机上的主要缺陷是函数重入。

3.1 函数的重入带来效率下降

$\mu\text{C}/\text{OS-II}$ 的运行, 对于函数重入是有要求的, 其系统运行所使用的很多系统函数都要求函数可以重入, 加上作为一种多任务抢占式内核, 函数在多任务

间的共用也要求函数可以重入。但是, 8051 构架的单片机, 通常硬件堆栈很小, 这就对函数的重入给出了限制。函数重入要求在函数被打断运行时系统能保存函数的形参、局部变量等运行环境, 函数的多次重入会使得堆栈需要保存的数据大量增加。任务切换、函数递归、中断嵌套等都会引起函数重入, 通常可以使用模拟堆栈的方式对硬件堆栈加以扩充, 可以在内存、外存中开辟一定的空间来模拟大容量的堆栈来保证函数重入的可靠执行。只是, 模拟堆栈虽然解决了硬件堆栈过小的问题, 同时却带来了系统运行效率的下降。在实际应用的过程中, 系统中函数重入带来的效率下降是很明显的, 尤其是在系统时钟比较慢的情况下, 情况会更加糟糕, 这也成了 8051 系统采用 $\mu\text{C}/\text{OS-II}$ 的一个瓶颈。

3.2 函数重入对实际应用的制约

函数重入降低系统效率, 理想情况是在具体应用时避免所有的函数重入。但这在现实应用中是不可能的, 在 $\mu\text{C}/\text{OS-II}$ 中, 很多系统函数本身就要求是可重入的, 否则就不能体现 $\mu\text{C}/\text{OS-II}$ 的多任务性和抢占性, 其实时性必然也就不能保证了, 所以 $\mu\text{C}/\text{OS-II}$ 系统函数是必须能重入的。能避免重入的最多也就是用户的任务函数了, 但是用户的任务函数如果不能重入, 对于程序设计是非常麻烦的, 在系统中如果有公共资源, 比如像键盘、显示屏、存储器等, 就不能在多任务间共享了, 这对系统的构架是一个严重的制约。

3.3 对函数重入缺陷的解决

用户任务如果经过精心的设计, 是可以做到尽量避免函数重入的。 $\mu\text{C}/\text{OS-II}$ 中任务间通讯可以有多种手段, 信号量、消息邮箱、消息队列等都可以用来在任务间传递消息, 可以利用这些手段来处理系统中的共享资源, 比如在系统中如果多个任务都要使用显示函数, 那就可以单独创建一个任务, 专门用来调用显示函数, 其他要使用显示函数的任务可以使用消息队列来给这个显示任务发送显示消息, 这样, 就可以保证多个任务共享显示函数, 但又不直接调用显示函数, 既然不会有多个任务直接调用显示函数, 那这个函数也就没必要设置成可重入了。当然这样的解决方案也不是唯一的, 但实践证明这种方案确实是可行

的, 在这种解决方案下, 对系统的软件构架有着严格的要求, 对程序员来说就多了一份约束。出于灵活考虑, 对于部分小型的、非常用的函数, 也可以采用一种效率较低的方案, 可以在调用这些函数时, 在函数的前后加上开关中断的语句, 这就保证了这些函数的调用不被打断, 在损失了部分抢占性的前提下保证了函数的不重入。对于递归函数, 前述方案并不能保证函数的不重入, 所以, 采用 $\mu\text{C}/\text{OS-II}$ 的 8051 系统不能使用递归函数, 否则可能会带来系统效率的巨大损失。

4 总结

8051 系列的单片机应用十分广泛, $\mu\text{C}/\text{OS-II}$ 是一种十分稳定可靠的抢占式嵌入式实时操作系统内核, 两者的结合可以解决很多实际的应用问题, 两者的结合也有着很多的优点, 但由于 8051 的体系结构问题, 两者的结合也有着一些缺陷, 这些缺陷可以被部分地解决, 避免缺陷的方式方法和实践应用的需求环境有很大的关系。

所以, $\mu\text{C}/\text{OS-II}$ 是否适合于 8051 系列的单片机要取决于实际的应用, 在大部分的情况下, 两者结合是有实际价值的, 很多的实践实例也证明了这一点; 在部分特殊的情况下, 比如系统时钟很慢, 对函数重入的系统开销十分敏感, 并且对系统实时性要求又十分严苛, 这时, 在 8051 系列的单片机上应用 $\mu\text{C}/\text{OS-II}$ 就不是光靠上文的相关方案可以解决的了, 好在 $\mu\text{C}/\text{OS-II}$ 本身代码开放, 8051 系列单片机选择范围又很大, 以后通过进一步的研究或许可以有较好的解决方式。

参考文献

- [1] 邵贝贝. 嵌入式实时操作系统 UC/OS-II[M]. 北京: 北京航空航天大学出版社, 2003.
- [2] 潘琢金, 施国君. C8051Fxxx 高速 SOC 单片机原理及应用[M]. 北京: 北京航空航天大学出版社.
- [3] C51 Compiler Users Guide [S]. Keil Elektronik GmbH. and Keil Software, Inc. 2001.
- [4] A51 Assembler User s Guide [S]. Keil Elektronik GmbH. and Keil Software, Inc. 2001.