[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/38

[12] 发明专利申请公开说明书

[21] 申请号 01126906.5

[43] 公开日 2003 年 4 月 9 日

[11] 公开号 CN 1409209A

[22] 申请日 2001.9.24 [21] 申请号 01126906.5

[71] 申请人 深圳市中兴通讯股份有限公司上海第二研究所

地址 200233 上海市桂林路 396 号

[72] 发明人 熊 勇

[74] 专利代理机构 深圳市顺天达专利商标代理有限公司

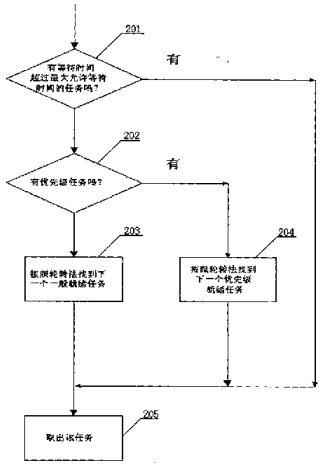
代理人 郭伟刚

权利要求书 2 页 说明书 11 页 附图 2 页

[54] 发明名称 一种多任务实时操作系统的实现方法

[57] 摘要

一种单片机嵌入式多任务实时操作系统的实现方法，为每个任务设置最大允许等待时间，将任务分为优先级任务和一般任务，用整数类型的数组记录每个任务的等待时间并在定时器程序中对其数值进行减量刷新；通过设置任务建立函数和挂起函数，将任务与其相对应的任务程序相连或在非临界区实现多任务调度，调度时，如果有等待时间超过最大允许等待时间的任务，则运行该任务；如果有优先级任务，则按照轮转法确定下一个就绪的优先级任务；如果没有优先级任务，按照轮转法确定下一个就绪的一般任务。 这种实现方法，允许多个任务并行，占用资源少，任务执行速度快，在满足实时性要求同时，也可提供足够的灵活性，调用也极为便利，针对性和实用性非常强。



ISSN1008-4274

1、一种多任务实时操作系统的实现方法，其特征在于：包括以下步骤：

（1）为每一个任务设置一个最大允许等待时间，将每一个需要占用 CPU 运行的任务分为优先级任务和一般任务，用一个整数类型的数组记录每个任务的等待时间并在定时器程序中对其数值进行减量刷新，其特征在于，包括以下步骤：

（2）设置一个任务建立函数，可将一个任务与其相对应的任务程序相连，并建立该任务的 TCB，将一个或多个其长度不大于基本单位时间的不定长时间片分配给一个任务；

（3）设置一个任务挂起函数，用于在非临界区的内存空间实现多任务之间的调度，调度时，用函数和全局变量的形式来替换掉调度处的局部变量，每一个任务在所分配到的时间片结束或被挂起之前占用 CPU；

（4）按照以下步骤进行多任务调度：保存现场状态、检查就绪任务队列中有无任务，如有任务则利用调度算法取出一个就绪任务，调用任务运行函数将该取出的任务设置为运行状态，恢复现场状态；其中，所述调度算法包括以下步骤：如果有等待时间超过最大允许等待时间的任务，则运行该任务；如果有优先级任务，则按照轮转法确定下一个就绪的优先级任务；如果没有优先级任务，按照轮转法确定下一个就绪的一般任务。

2、根据权利要求 1 所述方法，其特征在于，所述调用任务挂起函数是执行以下步骤：将第一参数表示任务号所对应的任务挂起第二参数表示的挂起时间，所述挂起时间以基本单位时间为单位，所述挂起时间为 0 时，将所述第一参数表示任务号对应的任务重新放到所述就绪任务队列的末尾；所述挂起时间不为 0 时，将所述第一参数表示任务号对应的任务挂起所述第二时间表示的挂起时间后，再重新放到所述就绪任务队列的末尾。

3、根据权利要求 1 所述方法，其特征在于，还包括以下步骤：设置任务就绪函数用于使某一个任务处于就绪状态。

4、根据权利要求 1 所述方法，其特征在于，还包括以下步骤：设置任务优先级设置函数，用于在调用该函数时，改变一般任务为优先级任务或改变优先级任务为一般任务。

5、根据权利要求 1 所述方法，其特征在于，还包括以下步骤：设置任务删除函数用于删除一个任务。

6、根据权利要求 1 所述方法，其特征在于，所述保存现场包括以下步骤：关中断；保存当前任务断点信息，如堆栈、程序指针、寄存器等到该任务的 TCB 环境中以及开中断；所述恢复现场包括以下步骤：关中断；用保存在该 TCB 中的环境信息来设置 CPU 堆栈、程序指针、寄存器等以及开中断。

一种多任务实时操作系统的实现方法

技术领域

本发明涉及操作系统设计与实现技术领域,具体涉及一种在单片机嵌入式实时操作系统中实现多任务调度的方法。

背景技术

当今，在以网络为中心的新兴计算设备和嵌入式市场中，嵌入式实时操作系统（RTOS）已经获得了广泛应用，尤其在 DSP、 PDA 和无线应用等。嵌入式 RTOS 孕育着巨大的商机。为降低成本，许多嵌入式应用都广泛采用各式各样的单片机来实现产品的智能化。单片机目标系统软件以前多采用汇编语言来实现。在 Franklin、Archimedes C51 等单片机 C 编译器产生后，由于 C 语言有丰富的库函数，编译效率高，用之来编写目标系统软件会大大缩短开发周期，增加软件的可读性，便于改进和维护。现在已经普遍采用 C 语言来开发 8051 等单片机了。但应用系统多种多样，应用软件也千差万别；如果设计一个单片机的 RTOS，那么在此 RTOS 的支持上，应用软件的编写就可以尽量简单，模块化，同时实现更好的可移植性。

在应用系统中，当目标系统软件比较复杂、规模比较庞大时，就需要一个操作系统的支持。目前，嵌入式的 RTOS 风靡全球，例如

Palm，VxWorks，Windows CE 等等。但购买一个商用 RTOS 成本太高，而且无法获得源代码，同时由于商用 RTOS 具备较完善的功能，规模较大，需要的内存和外存较多，较难移植到具体的单片机小系统上。因此编写一个适用性强，功能合适，规模较小的单片机操作系统就显得非常有必要了。这也有助于单片机应用软件的编写规范化和模块化。

发明内容

本发明要解决的技术问题是如何在单片机上实现微内核实时操作系统中，特别是一种在单片机嵌入式实时操作系统中实现的多任务调度方法。

本发明的技术问题是这样解决的，构造一种可用于单片机嵌入式系统的多任务实时操作系统的实现方法，其特征在于：包括以下步骤：为每一个任务设置一个最大允许等待时间，将每一个需要占用 CPU 运行的任务分为优先级任务和一般任务，用一个整数类型的数组记录每个任务的等待时间并在定时器程序中对其数值进行减量刷新；设置任务建立函数，可将一个任务与其相对应的任务程序相连，并建立该任务的 TCB，将一个或多个其长度不大于基本单位时间的不定长时间片分配给一个任务；设置任务挂起函数，用于在非临界区的内存空间实现多任务之间的调度，调度时，用函数和全局变量的形式来替换掉调度处的局部变量，每一个任务在所分配到的时间片结束或被挂起之前占用 CPU；按照以下步骤进行多任务调度：保存现场

状态、检查就绪任务队列中有无任务，如有任务则利用调度算法取出一个就绪任务，调用任务运行函数将该取出的任务设置为运行状态，恢复现场状态；其中，所述调度算法包括以下步骤：如果有等待时间超过最大允许等待时间的任务，则运行该任务；如果有优先级任务，则按照轮转法确定下一个就绪的优先级任务；如果没有优先级任务，按照轮转法确定下一个就绪的一般任务。

在按照本发明提供的方法中，所述调用任务挂起函数是执行以下步骤：将第一参数表示任务号所对应的任务挂起第二参数表示的挂起时间，所述挂起时间以基本单位时间为单位，所述挂起时间为 0 时，将所述第一参数表示任务号对应的任务重新放到所述就绪任务队列的末尾；所述挂起时间不为 0 时，将所述第一参数表示任务号对应的任务挂起所述第二时间表示的挂起时间后，再重新放到所述就绪任务队列的末尾。

在按照本发明提供的方法中，还包括以下步骤：设置任务就绪函数用于使某一个任务处于就绪状态。

在按照本发明提供的方法中，还包括以下步骤：设置任务优先级设置函数，用于在调用该函数时，改变一般任务为优先级任务或改变优先级任务为一般任务。

在按照本发明提供的方法中，还包括以下步骤：设置任务删除函数用于删除一个任务。

在按照本发明提供的方法中，所述保存现场包括以下步骤：关中断；保存当前任务断点信息，如堆栈、程序指针、寄存器等到该任务

的 TCB 环境中以及开中断; 所述恢复现场包括以下步骤：关中断；用保存在该 TCB 中的环境信息来设置 CPU 堆栈、程序指针、寄存器等以及开中断。

实施本发明提供的在单片机嵌入式实时操作系统中实现的一种多任务实时操作系统的实现方法，与许多现有 RTOS 相比，具有以下显著进步：允许多个任务并行，任务执行速度快，能够满足实时性要求；可支持足够多的并行任务，并行任务数量只受到实际硬件条件限制，例如 8051 系统内部和外部 RAM 空间。通过合理地调整程序的嵌套调用，可以允许 32 个或者更多的任务同步执行。这对于一般的嵌入式应用足够了；利用本发明方法实现似的操作系统占用极少的资源；并方便用户调用，对用户而讲，他无须了解 OS 的原理，不需要了解信号量等保证机制。只了解几个调用函数的用法即可，同时又给用户程序提供了足够的灵活性，在任一个合适的程序处都可以实现任务调度；针对性和实用性非常强。基于本发明的方法，可以在当前的许多软件增加 RTOS 的功能，而且整个过程非常简单，与此同时，还提供了一个 OS 平台，方便软件的模块化工作。

附图说明

图 1 是利用本发明方法进行任务调度的实现流程图；

图 2 是说明按照本发明方法的任务调度算法的实现过程的流程图。

具体实施方式

为叙述方便，以 8051 单片机作为以下实施例说明所依赖的硬件基

础。

为在嵌入式操作系统中实现多任务，同时还要满足实时性，需要对 CPU 时间进行划分。在本发明中，任务的时间划分不采用固定的时间片，而采用不定长时间片，时间片长短可以由用户任意设置，只要有足够精确的定时器。由于 8051 资源有限且应用系统千差万别，操作系统没有也不可能占用某一个具体的定时器，因此该定时器需要用户提供。假设用户设置了 8051 的定时器 T0，定时周期为 10ms，并且把该时长（10ms）作为任务调度的基本时间单位。那么，10ms 就是一个任务的最小时间片，也就是执行任务可能的最小延时时间。为了保证实时性，就需要用户任务程序在 10ms 内完成，如果无法完成，必须挂起该任务，腾出 CPU 给其他的需要运行的任务。

假设一个实例中有一个任务 task0，该任务全部执行完毕需要 15ms。那么，可以把该任务程序分成两部分。先执行该任务的第一部分约 8ms 的程序部分（只要小于 10ms 即可），挂起该任务后，允许其他任务占用 CPU，等待调度程序再次分配 CPU，来执行剩下的第二部分的程序。

那么任务是如何调度的？通常在多任务环境中，由于任务必须在自己已经分到的内存空间正确运行，互不干扰，互不破坏，这样对于每个任务而言，都需要足够的内存空间来保存当前的大量环境状态，这样操作系统占用的空间将会较大。如果一个用户程序中有 10 个任务，一个任务占用 1K RAM 空间，那么 10 个任务将占用 10K RAM ，这将消耗嵌入式用户大量的宝贵资源，非常不经济，适用性很差。

如前所介绍，本发明采用的多任务实现方法，实际上是类似于MSDOS，Win31 的“协作多任务”，即任务（程序）一直运行到它们主动让出 CPU 为止。而不是 Linux，Windows NT 所支持的“抢占调度多任务”。采用目前这种方法，显然用户程序需要在一个不处于临界区的地方（即是释放掉临界资源后的地方）实现任务调度，用函数和全局变量的形式来替换掉调度处的局部变量。这样，操作系统就不用保留各个任务当前的大量状态信息，并且不需要信号量，邮箱等复杂并发保证机制。这样，此操作系统就可以做到非常短小精悍，目前只占用 128 字节的 RAM 和 2k 的 ROM 空间（可以允许 8 个任务同时执行，如果任务增减，RAM 空间将比例增减。实际上，RAM 空间也就是所有 TCB 占用的空间），能够在一般的嵌入式系统中得到广泛的应用。

任务调度的具体实现，可以用一个挂起任务的函数来实现。该函数有两个参数，参数 1 表示挂起的任务号，参数 2 表示挂起的单位时间。显然，任何一个任务可以挂起其他的任务，也可以自身挂起；当自身挂起时，即是让出了 CPU；当挂起时间为 0 时，即是把自身任务重新放到了就绪任务的队尾。然后执行调度算法，把 CPU 分配给一个即将运行的任务。

那么采用哪种调度算法？基本上采用环形调度法，即时间片轮转法。假设就绪任务一共有 4 个，采用“ 1—>2—>3—>4—>1” 的调度方法，以保证所有任务都能执行。换言之，每个任务使用一个时间片的 CPU 后，释放处理机给下一个排在就绪队首的任务后，自己返

回到就绪队列的队尾（或者干脆挂起若干时间，腾出时间给其他就绪任务）。对于优先级任务，比如中断级别的任务，或者频繁执行但执行时间又很短暂的任务，可以抢先执行，但同样遵循环形调度法则。用户程序中需要定义一个最大等待时间。当一个一般级别的任务由于被优先级抢先，导致在就绪队列中等待时间超过最大等待时间时，此一般任务将被无条件的最先执行。采用上述的调度方法后，就可以保证所有任务都可以被执行，又能够保证优先级的任务的实时性。

在作为本发明方法一个实施例的操作系统中，提供以下接口给用户，方便用户的灵活应用：1）任务建立函数：把一个任务与其相对应的任务程序相连，并建立该任务的 TCB；2）任务删除函数：删除一个任务；3）任务就绪函数：使某一个任务处于就绪态；4）任务优先级设置函数：改变一个任务的优先级；5）任务运行函数：立刻执行某一个任务；6）任务挂起函数：挂起某一个任务一定的时间；7）操作系统启动函数：启动运行。

另外，需要一个整数类型的数组。该数组元素的个数为任务的总数，元素为各个任务等待的单位时间的时长。用户需要在用户的定时器程序中对所有数组元素进行减量操作。操作系统需要各个任务等待时长（操作系统本身没有定时计数功能），才能进行正确的调度和控制。

在图 1 示出的基于本发明的多任务调度方法的实现流程中，在框 101 中，关中断，在框 102 中，保存当前任务断点信息，如堆栈、程序指针、寄存器等到该任务 TCB 环境中，在框 103 中，开中断；在

框 104 中，检查在就绪任务队列中，有无队列（队列是否为空），如果队列为空，则返回框 104 继续检查；如就绪任务队列不为空，进到框 105，在框 105 中，按照其流程在图 2 中给出的调度算法取出一个就绪任务，并进到框 106 中，在框 106 中，设置该任务为运行状态，再在框 107 框中关中断，在 108 框中，用保存在该 TCB 中的环境信息来设置 CPU 堆栈、程序指针、寄存器等，在框 109 中，开中断。

在图 2 示出的调度算法的流程图中，在框 201 中，检查有无等待时间超过最大允许等待时间的任务吗，如没有，则在框 202 中，检查有无优先级任务，如没有，则在框 203 中按照轮转法找到下一个就绪的一般任务，最后在框 205 中，取出该任务；如在框 202 中查出存在优先级任务，则在框 204 中，按照轮转法找到下一个就绪的优先级任务，并进到框 205 取出该任务；如果在框 201 中，检查到存在有其等待时间超过最大允许等待时间的任务，则直接到框 205 取出该任务。

下面以一个具体的实例程序来说明利用本发明方法实现 RTOS 进行多任务调度的实际用法。

运行硬件环境：80C552 小系统； 显示：16×2 字符 LCD ；键盘：4 键。程序完成如下功能：LCD 左上角部分被两个显示任务轮流占用，LCD 右上角显示当前时间；LCD 左下角部分由一个任务实现动画显示，LCD 右下角显示一个实时数据，当有键盘按下时，该数据被改变。

简单说明一下该实例程序。该程序共有 6 个任务，分述如下：

Task0 每 100ms 执行一次，实现一个普通数据变量的累加，并且

显示到显示屏的坐标（0，0）处；

Task1 每 1 秒钟执行一次，读取并显示当前时间到坐标（8，0）处，同时在 1 分钟的时间内，允许 Task0 和 Task4 交替执行 30 秒；

Task2 实现简单的动画显示，显示到坐标（0，1）处；

Task3 每 500ms 实现一次实时数据的刷新，显示位置（8，1）；

Task4 实现简单的动画显示，显示到坐标（0，0）处；

Task5 每 10ms 读取键盘口，当有键盘按下时，改变实时数据，并且立刻运行 Task3。

显然，这里刻意把 Task0 和 Task4 显示内容处于同一个位置，来验证任务的删除和重建功能。

用户程序中设定了一个 10ms 定时器 T0，在定时器中断程序中对任务计数器数组中的每个值都运行一次减量运算（或者称为减一操作）。这样操作系统就可以根据任务计数器数组中的数值进行正确的任务调度和处理了。因此操作系统的时间单位就是 10ms。最快的响应和控制将在 10ms 后执行，从而满足实时性。

主要程序可以如下实现：

Task0：

首先，该普通数据变量递加，按十进制格式显示在坐标（0，0）处，然后自身挂起 10 个单位的时间即可。

Task1：

从一个实时钟芯片，如 DS1687 读取时间后，按时间显示格式显示在坐标（8，0）处；判断该时间是否为前 30 秒钟，如果是，重

建 Task0，删除 Task4，否则重建 Task4，删除 Task0，最后自身挂起 100 个单位的时间；

Task2：

在坐标（0，1）到（7，1）之间，显示一个位置循环左移的小图标，每次显示间距 100 毫秒，即自身挂起 10 个单位的时间；

Task3：

按十进制格式显示一个实时数据到坐标（8，1）处，然后自身挂起 50 个单位的时间；

Task4：

在坐标（0，0）到（7，0）之间，显示一个位置循环右移的小图标，每次显示间距 100 毫秒，即自身挂起 10 个单位的时间；

Task5：

读键盘口数据，如果没有键被按下，且持续 60 秒钟的话，关闭液晶背光；否则根据按键改变实时数据的数值，然后立刻调用 Task3（即完成数据的实时刷新），最后自身挂起 1 个单位的时间。

在用户的主程序中，开始分别调用任务建立函数来建立 6 个任务；然后初始化定时器 T0，最后调用操作系统启动函数即可。

另外，本程序中的 Task5 定时键盘处理任务摘自一个实际的监控系统软件中的 10 毫秒定时中断程序部分。从中可见，该任务每 10ms 执行一次，读取键盘口，消抖动，允许用户的加速键应用，每 30 秒自动送入一个退出键值到键盘缓冲区中。显然，该程序比较大而复杂，在定时中断程序中执行会降低软件可靠性和效率（中断程序应该尽量

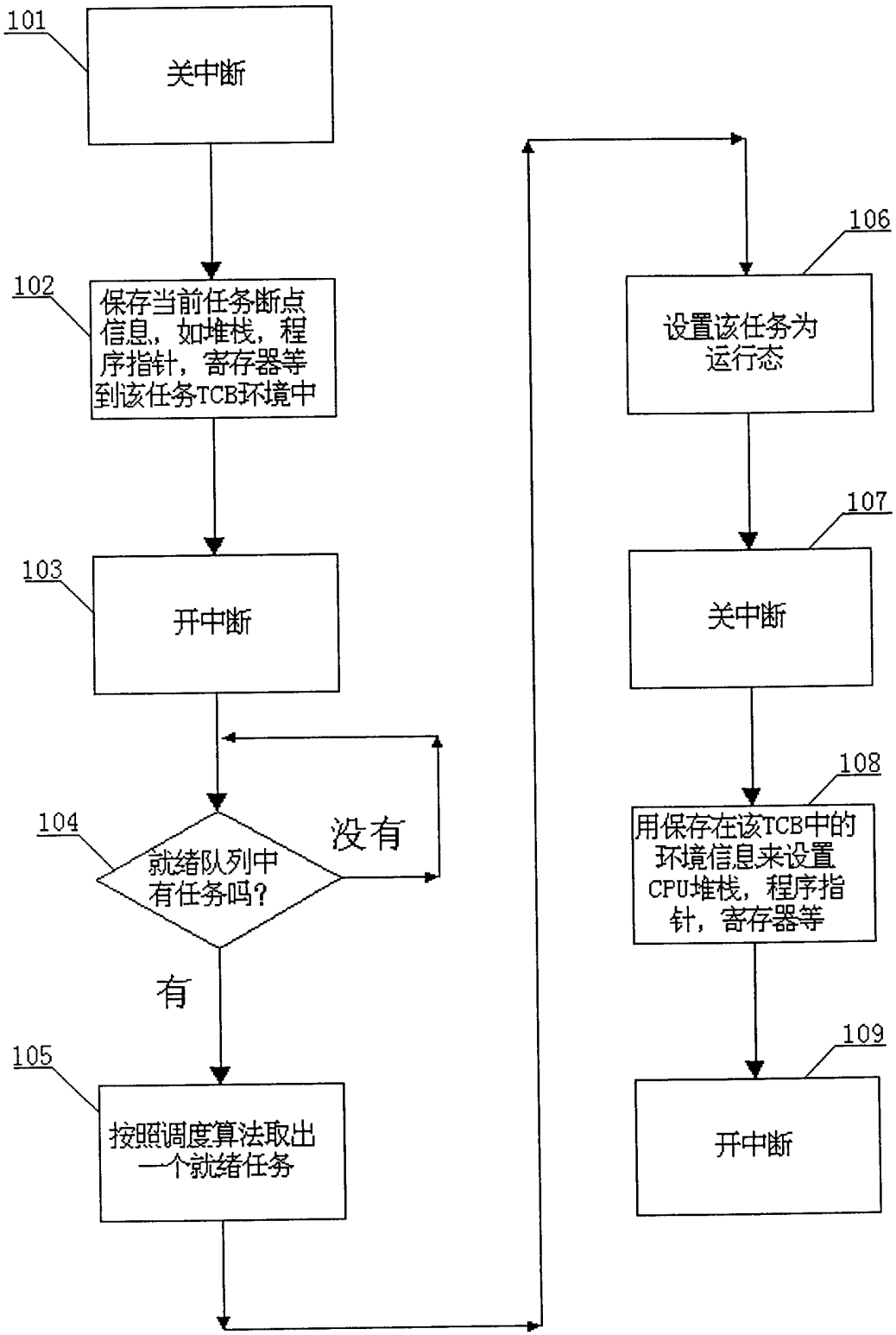

101
关中断
102
保存当前任务断点信息，如堆栈，程序指针，寄存器等到该任务TCB环境中
103
开中断
104
就绪队列中有任务吗?
没有
有
105
按照调度算法取出一个就绪任务
106
设置该任务为运行态
107
关中断
108
用保存在该TCB中的环境信息来设置CPU堆栈，程序指针，寄存器等
109
开中断

图 1

短小），在背景程序中执行又无法保证读键和显示的实时性。如上所述，把键盘处理函数当作一个任务执行，就完美地解决了上述问题。

综上所述，实现本发明提出的支持多任务调度的嵌入式 RTOS 的实现方法，在充分保证多个任务实时的同时保证响应的实时性。虽然上述实施例以单片机 8051 举例，实际上它可应用到其他类型的单片机系统上。它可以作为一种微内核 RTOS，实际应用时需要用户合理设置任务调度。由于无须开辟巨大的 RAM 空间来保存各个任务的环境，因此显得非常短小精悍，可以用在许多嵌入式应用中。运用之妙，在乎各人了。

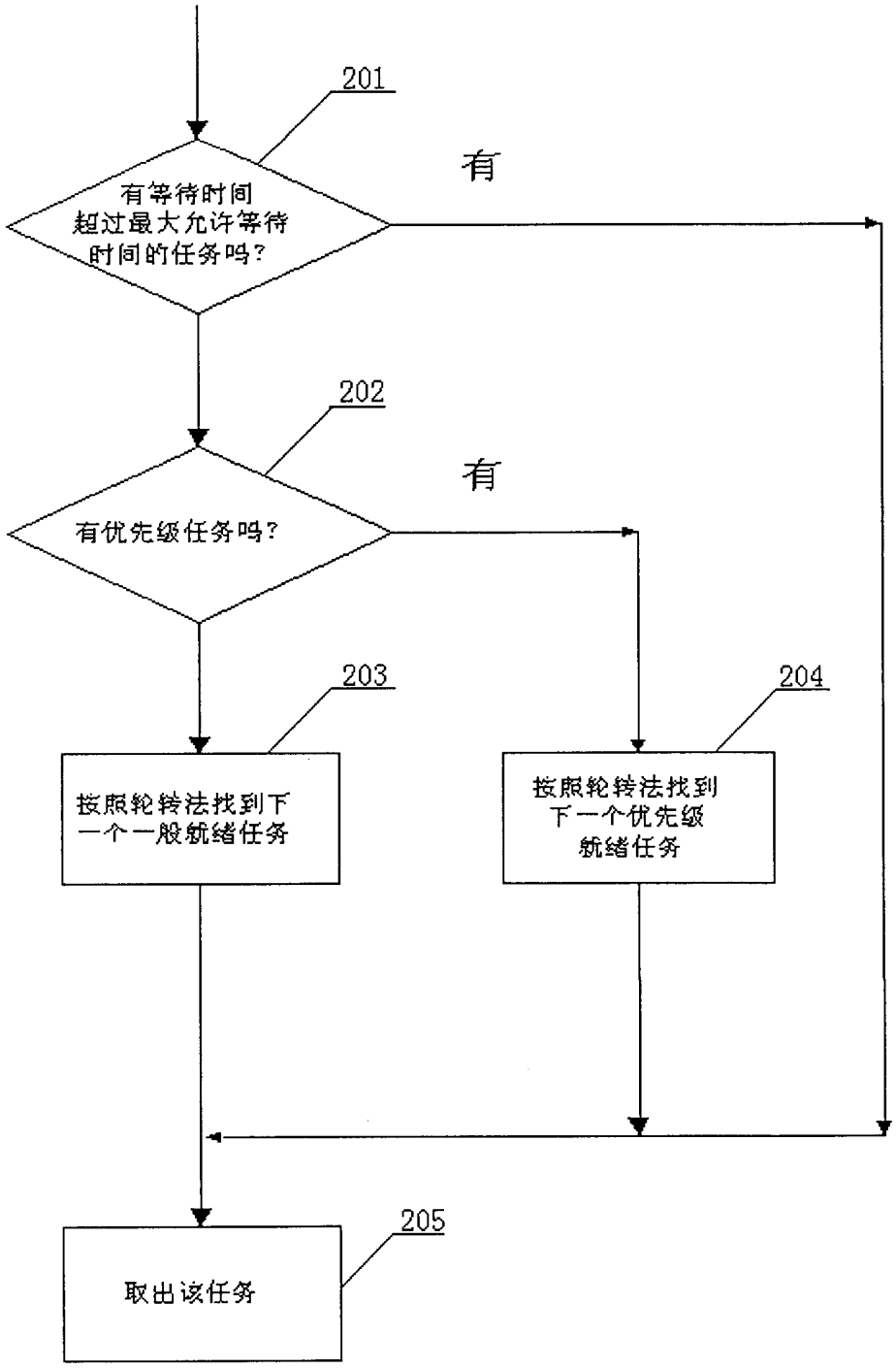

201
有等待时间
超过最大允许等待
时间的任务吗?
有
202
有优先级任务吗?
有
203
按照轮转法找到下
一个一般就绪任务
204
按照轮转法找到
下一个优先级
就绪任务
205
取出该任务

图 2