第32卷 第23期 计 算 机 工 程 2006年12月
Vol.32 No.23 Computer Engineering December 2006
·软件技术与数据库· 文章编号：1000—3428(2006)23—0085—03 文献标识码：A 中图分类号：TP316.2

μC/OS-II 内核结构分析及多任务调度实现

沈金荣，刘 翔
(河海大学常州校区计算机及信息工程学院，常州 213022)

摘 要：对嵌入式操作系统的结构、设计、用户界面等诸多方面进行深入研究，有助于嵌入式系统的应用和发展。该文从嵌入式实时操作系统μC/OS-II 内核入手，剖析了该操作系统的任务管理机理，结合工程应用实例，给出了多任务调度的实现过程。
关键词：实时操作系统；μC/OS-II；内核结构；任务管理；多任务

Analysis of μC/OS-II Kernel and Realization of Multi-task Schedulers

SHEN Jinrong, LIU Xiang
(College of Computer & Information Engineering, Hohai University, Changzhou 213022)

【Abstract】 Lucubrating the frame, the design and the user interface of the embedded system will do great help to the development and the implication of the embedded system. The passage takes μC/OS-II kernel of RTOS for example, analyses the task arrangement mechanism of the OS, and together with example of the project, and shows the process of the multi-task scheduler realization.
【Key words】 RTOS; μC/OS-II; Kernel configuration; Task management; Multi-task

嵌入式操作系统与嵌入式系统密不可分。嵌入式系统主要由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户的应用程序等 4 个部分组成，它是集软硬件于一体的可独立工作的“器件”。

μC/OS-II 是专门为计算机的嵌入式应用而设计的实时操作系统，是基于静态优先级的抢占式（preemptive）多任务实时内核。该内核不大，但具有一般多任务实时操作系统内核所具有的如可移植、可固化、可裁剪、可剥夺之类的全部特性。比较适合嵌入式系统的程序员、策划人员以及对实时操作系统有兴趣的学生。而且因为其经过了严格的测试，目前已被应用到了各个领域，如照相机业、发动机控制、网络接入设备、高速公路电话系统、ATM 机和工业机器人等。

内核是操作系统的核心部分，内核不能建立新的任务，它提供的基本服务是任务切换，此外还负责进程调度，消息处理，任务状态的转换等操作。如同一个公司的部门经理，不负责生产，只负责公司人员的调度、资源的分配等工作一样。多任务系统中，内核负责管理各个任务，或者说为每个任务分配 CPU 时间，并且负责任务之间的通信。内核结构的好坏直接关系到操作系统的效率，对实时操作系统而言，更关系到实时性问题。

1 内核结构

与其他商业内核比较，μC/OS-II 的内核结构较简单，但算法简单、结构紧凑、实时性较好。μC/OS-II 的内核结构包括任务控制块的结构、就绪表的结构、任务调度以及任务切换机理等，它们根据时钟节拍相互协调工作。

正确和全面理解内核结构，才能在具体的工程中缩短项目开发周期。这需要了解一系列关于实时系统的概念。其中对任务优先级、任务的理解是最基本的，也是最重要的。

μC/OS-II 是基于静态优先级的多任务实时内核。这决定了它进行任务管理等操作时，是以任务的优先级为基准的。每个任务都有其优先级。任务越重要，赋予的优先级越高。所谓静态优先级是指应用程序在执行过程中，诸任务的优先级不会改变。在静态优先级系统中，诸任务以及它们的时间约束在程序编译时均是已知。μC/OS-Ⅱ可以管理多达 64 个任务，μC/OS-Ⅱ 的 V2.52 版本有两个任务已经被系统占用了。作者保留了优先级为 0、1、2、3、OS_LOWEST_PRIO-3、OS_LOWEST_PRI0-2 ， OS_LOWEST_PRI0-1 以及 OS_LOWEST_PRI0 这 8 个任务以备将来使用。OS_LOWEST_PRI0 是作为定义的常数在 OS_CFG.H 文件中用定义常数语句 #define constant 定义的。因此用户可以有多达 56 个应用任务。必须给每个任务赋以不同的优先级，优先级可以从 0 到 OS_LOWEST_PR10-2。优先级号越低，任务的优先级越高。μC/OS-Ⅱ总是运行进入就绪态的优先级最高的任务。

一个任务，也称作一个线程，是一个简单的程序，程序认为 CPU 完全只属于该程序自己。应用程序的设计过程，包括如何把问题分割成多个任务，每个任务都是整个应用的某一部分，每个任务被赋予一定的优先级，有它自己的一套 CPU 寄存器和自己的栈空间。多任务运行的实现实际上是靠 CPU(中央处理单元)在许多任务之间转换、调度。

2 任务控制块结构

任务控制块（OS Task Control Block），是一个数据结构，在μC/OS-II 中用 OS_TCB 来表示。在任务被建立时，一个任务控制块 OS_TCB 就被赋值。当任务的 CPU 使用权被剥夺

作者简介：沈金荣(1970－)，男，讲师，主研方向：大学生创新教育实践，学生教育管理，计算机测控技术，自动控制系统应用等；刘 翔，助教
收稿日期：2006-08-09 **E-mail：**shenjr@webmail.hhuc.edu.cn

时，μC/OS-Ⅱ用它来保存该任务的状态。当任务重新得到CPU使用权时，任务控制块能确保任务从当时被中断的那一点丝毫不差地继续执行。

任务可以在多任务调度开始前调用函数 OSTaskCreate()或 OSTaskCreateExt()建立，也可以在其它任务的执行过程中调用这两个函数被建立。在 OSTaskCreate()中，程序调用了另外一个重要的函数 OS_TCBInit()。OS_TCBInit()首先试图从 OS_TCB 缓冲池中得到一个任务控制块 OS_TCB。如果缓冲池中有空余的 OS_TCB，这个 OS_TCB 就被初始化了。图 1 显示了 OS_TCB 被初始化后的变量与数据结构。由图 1 还可以看到，与任务控制块有关系的还有两个数据结构：OSRdyGrp 和 OSRdyTbl，即所谓的就绪表。

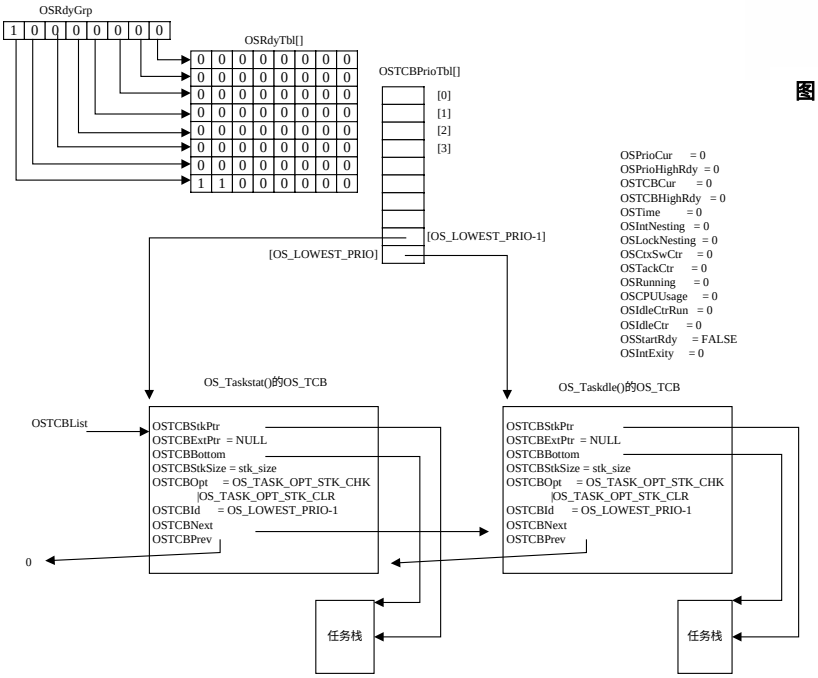

图 1 OS_TCB 被初始化后的变量与数据结构

3 就绪表结构

在μC/OS-Ⅱ中，每个任务被赋于不同的静态优先级，从 0 级到最低优先级 OS_LOWEST_PRIO，包括 0 和 OS_LOWEST_PRIO 在内。

就绪表中存放着每个就绪了的任务，等待内核调度器根据优先级进行任务调度。就绪表中有两个变量 OSRedyGrp 和 OSRdyTbl[]。在 OSRdyGrp 中，任务按优先级分组，8 个任务为一组。OSRdyGrp 中的每一位表示 8 组任务中每一组中是否有进入就绪态的任务。任务进入就绪态时，就绪表 OSRdyTbl[]中的相应元素的相应位也置位。就绪表 OSRdyTbl[]数组的大小取决于 OS_LOWEST_PR1O。当用户的应用程序中任务数目比较少时，减少 OS_LOWEST_PR1O 的值可以降低μC/OS-Ⅱ对 RAM（数据空间）的需求量。

OSRdyGrp 和 OSRdyTbl[]之间的关系见图 2，且满足以下规则：(1)当 OSRdyTbl[0]中的任何一位是 1 时，OSRdyGrp 的第 0 位置 1；(2)当 OSRdyTbl[1]中的任何一位是 1 时，OSRdyGrp 的第 1 位置 1；(3)当 OSRdyTbl[2]中的任何一位是 1 时，OSRdyGrp 的第 2 位置 1；(4)当 OSRdyTbl[3]中的任何一位是 1 时，OSRdyGrp 的第 3 位置 1；(5)当 OSRdyTbl[4]中的任何一位是 1 时，OSRdyGrp 的第 4 位置 1；(6)当 OSRdyTbl[5]中的任何一位是 1 时，OSRdyGrp 的第 5 位置 1；(7)当 OSRdyTbl[6]中的任何一位是 1 时，OSRdyGrp 的第 6 位置 1；(8)当 OSRdyTbl[7]中的任何一位是 1 时，OSRdyGrp 的第 7 位置 1。

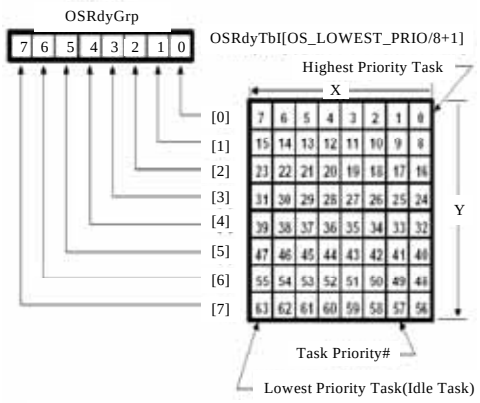

图 2 OSRdyGrp 和 OSRdyTbl[]之间的关系

下面的程序段完成使任务进入就绪态的工作：

```
OSRdyGrp |= OSMapTbl[prio >> 3] (1)
OSRdyTbl[prio >> 3] |=
OSMapTbl[prio & 0x07]          (2)
```

prio 是任务优先级变量，μC/OS-Ⅱ支持的任务优先级数为 64，所以 prio 的第 7 位以及第 6 位均为 0。结合图 3，由式(1)可以看出 prio>>3 得到的是任务优先级的次 3 位的值，而式(2)可以得到优先级的低 3 位的值。

在就绪表中，存在着一个很巧妙的算法（关系）：结合图 3 和表 1 可以看出，任务优先级的低 3 位用于确定任务在总就绪表 OSRdyTbl[]中的所在位。接下去的 3 位用于确定是在 OSRdyTbl[]数组的第几个元素。OSMapTbl[]是在 ROM 中的屏蔽字，用于限制 OSRdyTbl[]数组的元素下标在 0 到 7 之间。

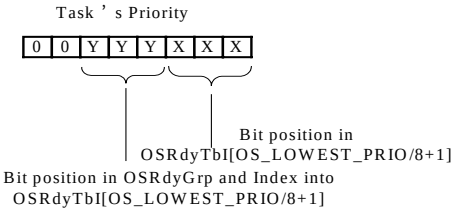

图 3 任务优先级变量比特位结构

表 1 OSMapTbl 的值

下标	位掩码（二进制）
0	00000001
1	00000010
2	00000100
3	0000100
4	0001000
5	0010000
6	0100000
7	1000000

任务调度器找出进入就绪态的优先级最高的任务，让其占有全部的 CPU 资源。如果要使一个任务脱离就绪态，则可用下面的程序进行处理：

```
if ((OSRdyTbl[prio >> 3] &= ~OSMapTbl[prio & 0x07]) == 0)
    OSRdyGrp &= ~OSMapTbl[prio >> 3];
```

以上代码将就绪任务表数组 OSRdyTbl[]中相应元素的相应位清零，而对于 OSRdyGrp，只有当被删除任务所在任务组中全组任务一个都没有进入就绪态时，才将相应位清零。也就是说 OSRdyTbl[prio>>3]所有的位都是零时，OSRdyGrp 的相应位才清零。

4 任务调度与任务切换机理

μC/OS-Ⅱ总是运行进入就绪态任务中优先级最高的那一个。确定哪个任务优先级最高，下面该哪个任务运行了的工作是由调度器（Scheduler）完成的。任务级的调度是由函数 OSSched()完成的。

在函数 OSSched()中，我们找出了最关键的程序，该程序用于找出进入就绪表的优先级最高的任务：

```
y    =   OSUnMapTbl[OSRdyGrp];
x    =   OSUnMapTbl[OSRdyTbl[y]];
prio = (y << 3) + x;
```

以下结合图 4 讲述寻找最高优先级任务的机理。

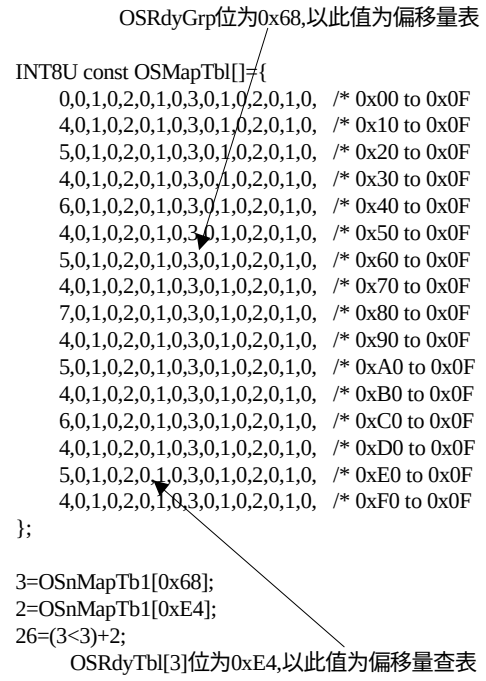

图 4 最高优先级任务

假设 OSRdyGrp 的值为二进制值 01101000，即 0x68，则查 OSUnMapTbl[OSRdyGrp]得到的值是 3，它相应于 OSRdyGrp 中的第 3 位 bit3，这里假设最右边的一位是第 0 位 bit0。类似地，如果 OSRdyTbl[3]的值是二进制 11100100，即 0xE4，则 OSUnMapTbl[OSRdyTbl[3]]的值是 2，即第 2 位 bit2。于是，任务的优先级 prio 就等于 26，即(3*8+2)。利用这个优先级的值，查任务控制块优先级表 OSTCBPrioTbl[]，得到指向相应任务的任务控制块 OS_TCB。

找到优先级最高的任务之后，OSSched 检验这个优先级最高的任务是否是当前正在运行的任务，以避免不必要的任务调度，因为对实时性要求很高的嵌入式系统设备而言，很浪费时间的任务调度能避免就应该避免。

任务切换很简单，由以下两步完成，将被挂起任务的微处理器寄存器推入堆栈，然后将较高优先级的任务的寄存器值从栈中恢复到寄存器中。在μC/OS-Ⅱ中，就绪任务的栈结构总是看起来跟刚刚发生过中断一样，所有微处理器的寄存器都保存在栈中。换句话说，μC/OS-Ⅱ运行就绪态的任务所要做的一切，只是恢复所有的 CPU 寄存器并运行中断返回指令。为了做任务切换，运行 OS_TASK_SW()，人为模仿了一次中断。多数微处理器有软中断指令或者陷阱指令 TRAP 来实现上述操作。中断服务子程序或陷阱处理（Trap hardler），也称作事故处理（exception handler），必须提供中断向量给汇编语言函数 OSCtxSw()。OSCtxSw()除了需要 OS_TCBHighRdy 指向即将被挂起的任务，还需要让当前任务控制块 OSTCBCur 指向即将被挂起的任务。

5 多任务调度的实现

笔者在承接的一个项目汽车行驶记录仪中使用μC/OS-Ⅱ进行了多任务的管理。

项目要求记录仪能在汽车行驶中完成检测汽车是否启动检测，汽车是否停止检测，8 路开关量，4 路门信号量，行驶速度检测，行驶里程检测，连续驾驶时间检测，通过 USB 传送数据，打印机打印数据，菜单操作，液晶显示等操作。笔者建立了 11 个任务，根据任务的紧迫性，为其分配的优先级如表 2 所示。

表 2 任务优先级分配表

任务名称	任务	优先级
汽车是否启动检测	VTDRCStTest	4
汽车是否停止检测	VTDRCSpTest	5
8 路开关量检测	VTDRESTest	6
4 路门信号量检测	VTDRFSTest	7
行驶速度检测	VTDRRSTest	8
行驶里程检测	VTDRRMTest	9
连续驾驶时间检测	VTDRCRTTest	10
通过 USB 传送数据	VTDRUSBTran	11
打印机打印数据	VTDRPrint	12
液晶显示	VTDRLcm	13
菜单操作	VTDRMenu	14

汽车是否启动检测任务 VTDRCStTest 的优先级最高。因为记录仪上电后首先要执行的是这一步操作。汽车是否停止检测任务的优先级次之，因为，如果汽车停止了，诸如行驶速度检测之类的任务就可以被挂起了。其它的任务优先级仅仅按照先后顺序依次随机分配，其实它们之间的优先级应该是相同的。

当任务 VTDRCStTest 执行，且检测到汽车已经启动后，调用函数 OSTaskSupend()将自身挂起。在汽车停止之前，不再让这个任务进入就绪态。因为既然汽车启动检测工作已经完成，而且检测到汽车已经启动后，在汽车停止前这一步就没有必要再执行了。

当汽车是否停止检测任务 VTDRCSpTest 检测到汽车停止后，OSTaskResume()将汽车是否启动检测任务恢复。其它任务每次执行完都会调用函数 OSTimeDly()将自身延时，暂时退出就绪态。延时时间不等。

6 结束语

嵌入式操作系统与普通系统相比是更为健壮的、低成本的、特性完备的操作系统。它的应用将大大提高嵌入式系统开发的效率，改变以往嵌入式软件设计只能针对具体的应用从头做起的状况，在嵌入式系统之上开发嵌入式系统将减少系统开发的工作量，增强嵌入式应用软件的可移植性，使嵌入式系统的开发方法更具科学性。

μC/OS-II 作为一个嵌入式操作系统内核已被广泛应用在

（下转第 113 页）