实用嵌入式操作系统 OS20内核分析

黄 浩，徐定翔

（华中科技大学计算机科学与技术学院信息存储系统教育部重点实验室，武汉 430074）

摘 要：介绍了 ST公司的嵌入式实时操作系统 OS20，指出其基本特点和常用范围，着重分析了其主要内核功能模块，即任务管理，时钟管理，内存管理，中断管理，信号量管理和消息管理，并举例说明了其应用。

关键词：OS20；实时操作系统（RTOS）；嵌入式；内核

中图分类号：TP31 文献标识码：A 文章编号：1002－2279（2008）03－0177－03

Kernel Analysis of the Practical Embedded Operating System OS20

HUANG Hao, XU Ding-xiang

(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: This paper introduces the embedded real time operation system OS20 of ST company. Analyzes the major kernel modules of OS20, such as tasks schedule, timing, memory management, interrupts, semaphores and message queues managements at length. Besides, its basic features and typical applications are presented.

Key words: OS20; Real Time Operating System; Embedded System; Kernel

1 引 言

随着信息技术（IT）的飞速发展，特别是互联网的迅速普及，3C（计算机、通讯、消费电子）合一的加速，数字化时代已经来临。嵌入式接入设备是数字化时代的一大主流产品，世界各国在此领域开始了激烈竞争，以争取获得主导地位。嵌入式软件是数字化产品的核心，作为嵌入式软件的基础和龙头，嵌入式操作系统在产业发展过程中扮演了越来越重要的角色。嵌入式操作系统的出现，大大提高了嵌入式系统开发的效率，改变了以往嵌入式软件设计只能针对具体应用从头做起的状况。在嵌入式操作系统之上开发嵌入系统减少了系统开发的工作量，增强了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。

2 STLite/OS20简介

OS20嵌入式操作系统全称为 STLite/OS20，是国际著名的意法半导体公司（简称 ST公司）为其生产的一系列嵌入式芯片编写的一个通用的实时、多任务操作系统。这些芯片涉及的应用泛围相当广泛，如用于数字电视接收的 STi5516、STi5518系列芯片，用于 GPS定位的 STB5610系列芯片等。在笔者参与设计的 DVB－T数字电视接收机项目开发方案中，就采用了 ST公司的 STi5516芯片及 STLite/OS20嵌入式操作系统。

3 OS20内核模块的功能介绍及其应用

STLite/OS20是一款嵌入式实时多任务操作系统，它具有以下一些基本特征：①采用基于优先级的多任务抢占式调度策略；②内核短小，实时性好，能进行快速的上下文切换（小于 5μs），效率高；③开放源代码，并且支持一系列的处理器芯片；④对内存要求低，对于内核，只要求 600B的 internal和 11K RAM，对于每个进程，只要求 24B的 internal和 36B的 external。

OS20提供了高效的多任务环境，包括任务调度管理、时钟管理、内存管理、中断处理、通过信号量实现同步和互斥、以及通过消息队列实现消息通信等等。以下分别对这几个重要模块进行详细的说明。

3.1 任务管理

OS20是多任务抢占式操作系统，此处的任务即指进程。在 OS20中，一个进程由数据结构、栈空间和代码段组成，进程在创建时必须指定优先级（可动态修改）并分配一定大小的栈空间。进程分为 16个优先级（0～15，15最高，如有必要可扩展至 64级），按

作者简介：黄浩（1965－），男，湖北武汉人，副教授，博士，主研方向：光信息存储技术、多媒体信息存储与处理。

收稿日期：2006－08－15

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

优先级从高到低依次调度，同优先级之间按时间片轮转调度，时间片大概在 1ms~2ms。程序开始启动时，先执行一个单独的主进程，通过主进程再依次创建其余的进程。这些进程可被认为独立于主进程之外运行，但是所有进程共享 CPU资源和内存空间。

对于栈空间的分配，分配多了会造成浪费，分配少了可能会出现栈溢出。为了避免出现上述情况，OS20采用了一种技术：初始化进程时分配较大的栈空间，运行进程并检测其实际使用的空间大小，再依据实际情况进行调整，以达到较优的栈空间分配。

操作系统对于任务的基本操作主要包括任务的创建、调度、暂停、休眠、重调度、终止和删除等等。由于优先级机制是任务调度管理的基础，下面就主要介绍 OS20的优先级原理。

OS20的任务优先级虽然在其进程被创建的时候就确定了，但在运行过程中可以通过调用 task_priority_set()来动态修改其优先级。如图 1所示，任务调度内核能够保证当前执行的任务总是最高优先级的，它总是一直运行到结束或者直到被更高优先级的任务所抢占（相同优先级的任务则分时运行）。一旦当前的最高优先级任务运行结束，次高优先级的任务才能被 CPU执行。

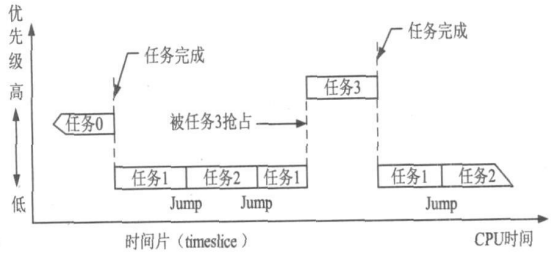

图 1 OS20的任务调度示意图

对于不想被分时或抢占的任务，可以采用下面这对函数

```
void task_lock(void);
void task_unlock(void);
```

实现进程锁定功能，让被锁定的任务独占 CPU的运行时间。

3.2 时钟管理

时钟是实时操作系统中不可缺少的一部分。OS20有 2个不同分辨率的片上实时时钟，分别为：低分辨率时钟，分辨率为 64μs，总计 38hours，为低优先级进程使用；高分辨率时钟，分辨率为 1μs，总计 30minutes，为高优先级进程使用。

图 2为 OS20的时间算法原理图，从图中可以看出，时间值的计数（ticks）范围是 0x00000000 ~ 0xFFFFFFFF，周期为 2^32。

例如，实时操作系统应用中的超时设置，可以通过以下代码实现：

```
startTime = time_now();
……
While(TRUE)
{ curTime = time_now();
if(time_minus(curTime, startTime) >= WAIT_FOR_1_SEC * 5)
{ update();
break;
}
else
…
}
```

其中，WAIT_FOR_1_SEC代表 1s时间的计数，上例实现的是 5s的超时。

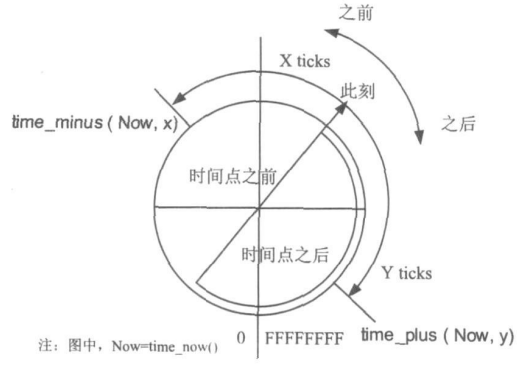

图 2 OS20时间算法图

3.3 内存管理

内存管理在嵌入式操作系统中是至关重要的，因为内存资源往往很有限，必须有效地利用。

OS20提供了三种不同风格的内存管理，通过控制内存分配的方式，使用户能够在平衡时间和空间的性能上有足够的灵活性。三种不同的内存分配方案如下：① Heap：任意大小分配，用完后须回收，需要 12B的系统开销；② Fixed：固定大小分配，用完后须回收，需要 4B的系统开销；③ Simple：任意大小分配，用完后不回收，不需要任何系统开销。

3.4 中断处理

OS20的中断处理系统，满足实时性要求：①中断一旦发生，完全由硬件负责迅速将 CPU切换到中断处理例程；②中断分 0~7共 8个级别（7级最高），高级别中断可以中断低级别中断的处理例程，实现中断嵌套，保证最重要的事情（实时性要求最强）总是能被最先处理；③通过一个 Interrupt Level Controller可将中断级别数扩展至 32个甚至 128个。

3.5 同步互斥管理

在多进程操作系统中，各进程之间的同步和对共享资源的互斥访问显得尤为重要，OS20使用最常用且有效的信号量来实现进程间的同步与互斥管理。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

3.5.1　OS20中信号量的基本概念

在 OS20中,一个信号量的数据结构包含两个域:①信号量能被使用的次数,当次数<=0时,不能被使用;②等待占有这个信号量的任务队列。OS20的信号量机制提供两个最基本的操作:等待信号量 semaphore_wait(mutex)和释放信号量 semaphore_signal(mutex),以此来实现各种复杂的功能。

semaphore_wait的操作过程相当于信号量的 P操作:若 mutex为正,则调用该函数的进程得到资源,继续运行,并将 mutex减 1;反之,若计数值为负,该进程将会被阻塞,直到获得此信号量为止。

semaphore_signal的操作过程相当于信号量的 V操作:若等待队列上有进程挂起,则启动队首的第一个进程,反之,将计数值加 1。

为适应不同的需要,有两种信号量可以使用:① FIFO Semaphore,等待某个信号量的各进程公平排队,先到先服务;② Priority Semaphore,优先级高的进程可以插队排到队首,从而能够先获得资源而先运行,这种模式的信号量常常用于内核进程当中。

3.5.2　用信号量实现任务对资源的同步、互斥访问

信号量主要用于任务同步和共享资源互斥访问,下面对这两个功能作简要介绍。

1)同步操作

通常用在一个任务等待某个中断 Handler,此时相关信号量初始化为 0,任务 2对其执行 semaphore_wait操作后就一直等待,直到中断处理进程任务 1对该信号量执行 semphore_signal后,任务 2才开始继续执行。

2)互斥进入

在"计数"模式下,相关信号量初始化为 N(N为同时进入共享资源的任务个数),若当前获得共享资源的任务数达到 N时,接下来调用 semaphore_wait的任务就会排在等待"队列"中(FIFO或 Priority),直到正在占用资源的某个任务退出(调用 semaphore_signal使信号量值加 1,释放可用资源),等待"队列"中的任务才能进入共享资源访问。

典型情况下,即在"二值"(即信号量的值只能是 0或 1)模式下,相关信号量初始化为 1,任何时候只能有一个任务进入共享资源访问。这在实际中是经常用到的。此时,用相关信号量保护的代码段区域被称为"临界区"。该信号量的值为 1时,表示临界区空闲;为 0则表示临界区被占用了,即其他任务不能再占用。

3.6　通信机制

在 OS20中,通过消息队列来实现任务间的通信。消息队列是任务间的一种缓冲通信方法,它避免了任务间大量通信数据的复制,使整个系统更高效、更省时。

3.6.1　消息元素的组成结构

消息队列在创建或初始化的时候就确定了其大小(字节数)和消息个数,其中单个消息元素结构如下:

```
typedef struct{
int tag;
char msg[MSG_SIZE];
}msg_t;
```

3.6.2　消息队列的工作原理

如图 3所示,某任务 A要发送消息之前,首先调用函数 message_claim()在要发往的有空闲的队列中申请一个消息缓冲,然后调用 message_send()将该消息发送到发送队列中。

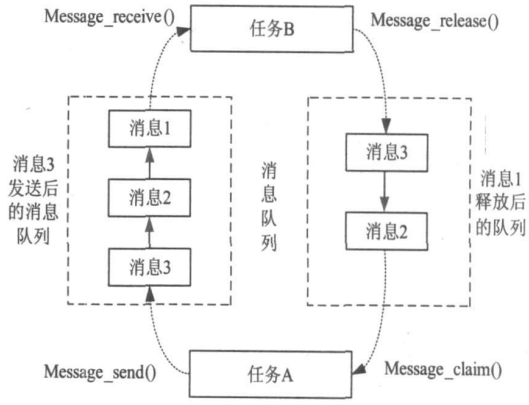

图 3　OS20的消息队列工作机制

任务 B要接收消息,调用函数 message_receive()接收刚才由函数 message_send()发送来的消息,并返回其指针,最后用此指针作参数调用 message_release()以释放由任务 A通过调用 message_claim()申请的消息缓冲空间。

可见,使用消息队列,不需要数据拷贝,就可实现不同任务之间的通信或数据传递。

4　结 束 语

STLite/OS20具有模块化、层次化的结构,非常便于软件移植、管理和功能升级。其内核短小,并可以根据用户需求进行裁剪(通过 C语言中的条件编译来实现)。随着 ST公司一系列嵌入式芯片的广泛应用,STLite/OS20也将被越来越多的嵌入式系统设计者所接受。

参考文献:

[1]　郑宗汉.实时系统软件基础[M].北京:清华大学出版社,2003.

[2]　杨勇,饶群.数字电视机顶盒的嵌入式操作系统内核分析[J].电视技术,2003(3):50-53.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net