RTX51 Tiny 任务切换的分析

苑广军[1] 胡冬梅[1] 孙继元[1] 任 辉[2]
(1 东北电力学院 2 昆山华恒公司)

摘 要 简要介绍了 RTX51 Tiny 的基本特性,任务状态及任务状态字结构;详细地分析了任务切换时的存储器管理,并给出任务切换的主要代码流程图。

关键词 RTX51 Tiny 实时操作系统 单片机 任务切换

Analysis of The Task Switching on RTX51 Tiny

Yuan Guangjun[1] Hu Dongmei[1] Sun Jiyuan[1] Ren Hui[2]
(*1. Northeast china Institute of Electric Power Engineering 2. Huaheng company of Kunshan*)

Abstract Essentialities of RTX51 Tiny ,the task states and the construction of task states are introduced simply, the memorizer manage is analyzed in detail when tasks are switched, and the flow chart of primary codes of the task switching is given in this paper.

Keywords RTX51 Tiny real - time operating system MCU task switching

1 RTX51 Tiny 简介

RTX51 是 KEIL 公司开发的一个适用于 8051 系列的实时多任务操作系统。RTX51 使复杂的系统和软件设计以及有时间限制的工程开发变得简单,它可以在单个 CPU 上管理几个作业(任务)。RTX51 Tiny 是 RTX51 Full 的一个子集,它可以很容易地运行在没有扩展外部存储器的单片机系统上 。

RTX51 Tiny 最多可定义 16 个任务,所有任务可同时激活,允许循环任务切换,仅支持非抢占式的任务切换,操作系统为每一个任务分配独立的堆栈区,并在任务切换时改变堆栈指针,保存和恢复寄存器的值。它大约占用 900 个字节 ROM,能并行的利用中断功能,用户无需编写源代码,系统会自动生成,但任务没有优先级和中断管理。RTX51 Tiny 没有独立的时间服务函数和使任务挂起的函数,而是使用 OS_wait( )通过设定不同的参数来实现的。OS _wait( )函数可以等待以下事件:时间到、时间间隔、来自任务或者中断的信号 。

2 任务的状态及状态字结构

2.1 任务的状态[1]

RTX51 Tiny 的内核为每个任务分配状态,使用 RTX51 Tiny 定义的每个任务必须处于下列的状态之一。状态描述如下:

RUNNING:任务正在运行中,每次只能有一个任务处于运行状态。

READY :任务正在等待运行,当前运行的任务完成后,开始运行处于等待中的任务。

WAITING:任务正等待一个事件,若发生的事件是任务所等待的,任务则进入"READY"状态。

DELETED:任务尚未启动。

TIMEOUT:任务被一个循环"时间到"所中断,与"READY"状态相似,但由于内部操作过程使一个循环任务切换而被冠以标记。以建立 4 个任务为例,表 1 为 RTX51 Tiny KEIL C51 主程序运行时的任务列表,创建任务 0 并启动,将各个任务堆栈栈底指针初始化,8052 默认的栈顶 (RAMTOP) 值为 FFH。此表显示了当前各个任务运行状态及堆栈分配情况:

表 1:RTX-TINY-TASKLIST

TID	Task Name	State	Wait for Event	Sig	Timer	Stack
0	init	Tunning		0	0X00	0X20
1	scankey	Deleted		0	0X00	0XFF
2	display	Deleted		0	0X00	0XFF
3	display1	Deleted		0	0X00	0XFF

2.2 任务状态字结构

由于任务有以上几种状态,所以要标记任务状态就必须设置状态字,任务状态字结构如下定义:

① 任务状态.0 = 等待信号

② 任务状态.1 = 等待时间到

③ 任务状态.2 = 信号标志

④ 任务状态.3 = 时间到标志

⑤ 任务状态.4 = 任务就绪(等待运行)

⑥ 任务状态.5 = 任务激活(OS_create 使能位)

⑦ 任务状态.6 = 轮转时间到

⑧ 任务状态.7 = 运行标志

3 任务切换

3.1 任务切换概念

RTX51 Tiny 内核提供的基本服务就是任务切换。简单地说,任务切换就是将全部自由堆栈空间分配给正在运行的任务。具体地,当多任务内核决定运行另外的任务时,保存正进行任务的当前状态,即保存 CPU 寄存器中的全部内容,这些内容保存在自己的堆栈区之中。进入堆栈工作完成后,就把下一个将要运行的任务的当前状态从任务的堆栈中重新装入 CPU 寄存器中并开始下一个任务的运行,这个过程称为任务切换。RTX51 Tiny 是基于时间片轮转调度算法的操作系统,它支持非抢占式的任务切换,即允许每个任务运行直到该任务自愿放弃 CPU 的控制权,中断可以打入运行着的任务,中断服务完成后将 CPU 控制权还给被中断的任务 。所以当任务主动放弃 CPU 时,RTX51 Tiny 才进行任务切换,这是一种情况;另一种情况是当一个任务的时间片用完时,进行任务切换。任务切换时 RTX51 Tiny 保护现场,进行系统堆栈管理,至少保护 15 个字节。

3.2 任务切换主要代码流程图

3.2.1 流程图

如图 1 所示:

由程序框图可以看出,任务切换有两个入口:定时时间到和等待或删除任务。相应地有两个出口:恢复保护现场的是前者出口,清一些状态标志位的是后者出口。在任务堆栈管理方面,为给当前正运行的任务分配尽可能大的栈区,就将下一个任务的栈区以读出压入方式(或弹出写入方式)移至自由空间的下端(或上端)。当任务切换开始时,根据任务所处状态的不同和有无事件发生,对任务进行相应的处理,以完成任务间的切换。

3.2.2 时间片(time slice)

RTX51 Tiny 执行循环任务切换,处理允许"准并行"执行多个无限循环或任务。各个任务并非持续运行,只在时间片(一段预先设定的时间)内才并行执行。CPU 执行时间被分为若干时间片,RTX51 Tiny 为每个任务分配一个时间片。每个任务只允许在预定时间内执行,然后,RTX51 Tiny 切换到另一任务并允许它在其规定的时间段内执行。时间片持续时间通过配置变量 TIMESHARING 来进行设置。

不必等待一个任务的时间段结束就可以采用系统函数 OS_ wait 来通知 RTX51 Tiny 允许另一个任务开始运行,函数将当前执行的任务挂起以等待一个特定事件的发生,在这段等待时间内可以运行任意数目的其他任务 。

3.2.3 临界段(Critical Section)

RTX51 Tiny 只能轮转服务于一系列任务中的某一个,当一个任务的时间片使用完时,才可进行任务切换。为避免同时有其他任务或中断服务进入,在任务切换延时服务程序中,即等待超时的时候,调

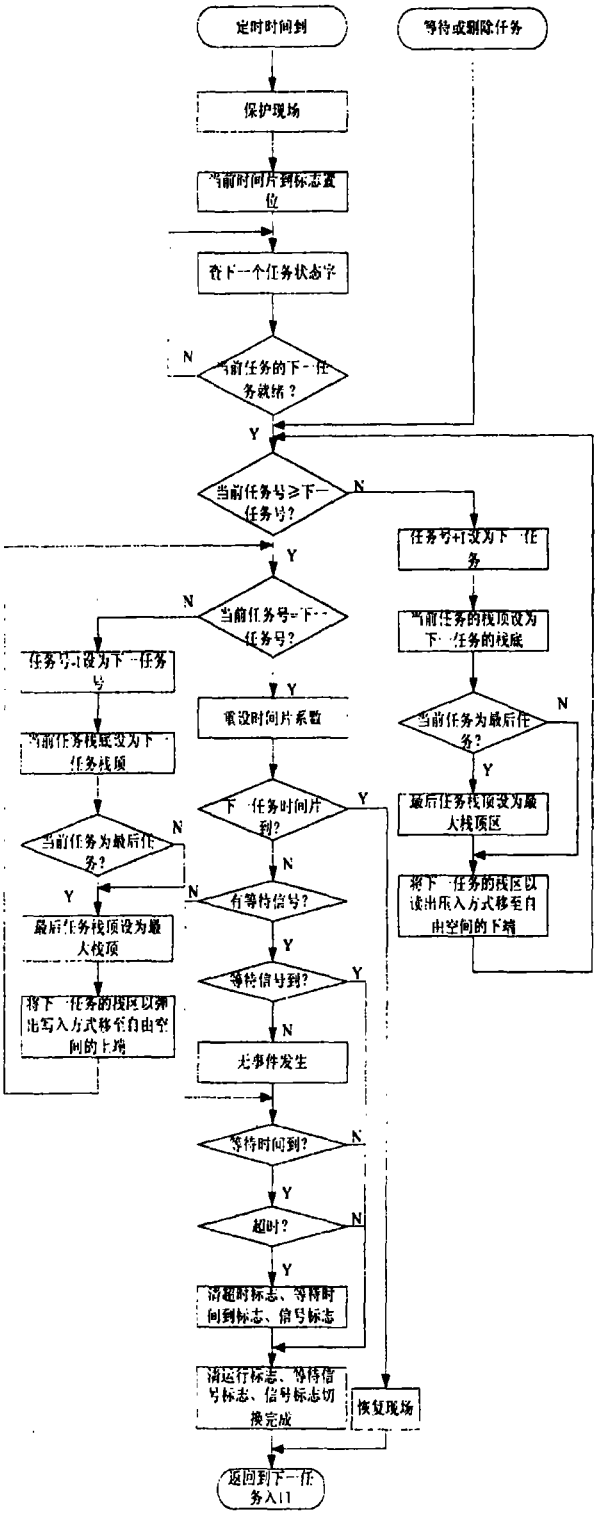

图 1

用 OS_ wait 的任务将延时一段时间。这段时间的长短用系统节拍数确定,判断指定节拍数是否为 0 的代码属临界区代码。如果指定节拍数为 0,则不执行延时。为了防止任务被程序意外变为就绪状态,使用一个循环判定延时时间是否到达;如果没有到达,则将任务置为等待状态,然后进行任务调度(task scheduling),执行下一个任务。

3.3 任务切换中的堆栈管理

KEIL C51 编辑器对变量使用静态分配存储空间,因此存储器管理简化为堆栈管理 。RTX51 Tiny 对内存分配大致分为如下三个部分:第一部分是当前任务寄存器(寄存器组 0:R0, R1, R2, R3 R6, R7)。第二部分是当前任务存储区。第三部分是两张表:其一是任务堆栈表,其二是任务状态表。RTX51 Tiny 用 8052 单片机的内部存储器为每个任务设定堆栈,并且为每个任务保留一个单独的堆栈空间。

自由堆栈空间字节大小由 FREESTACK 来定义,切换任务时,RTX51 Tiny 校验堆栈中的有效字节段,如果堆栈太小,RTX51 Tiny 就会调用宏 STACK_ERROR ,进入堆栈出错处理。字节数默认值为 20 个字节。

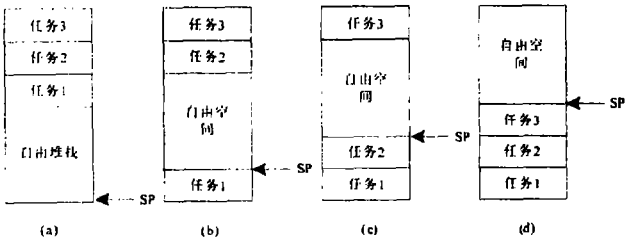

图 2

在执行任务 0 时,各任务栈区分布如上图所示。在栈底部是提供任务 0 使用的堆栈自由空间,而顶部是为各个任务的堆栈区。切换时要找出已经就绪的等待切换的任务,寻找是从当前任务号开始按照任务号的顺序进行的,如果下一个任务未被激活,则继续向下寻找一个被激活的任务。找到后,再比较当前任务号与待切换任务号大小,如果当前任务号小于待切换任务号,就执行栈区下移;如果当前任务号大于待切换任务号,就执行栈区上移。栈区移动采取读出压入或弹出写入方式。弹出写入(POP)方式,堆栈指针自动减,使栈区上移,上图指针移动后,待切换任务栈顶移至自由空间底部。读出压入(PUSH)方式,堆栈指针自动加,使等待切换的任务栈区下移,待切换任务栈顶移至自由空间顶部。通过这两种方式就给任务分配了尽可能大的栈区了。

如果当前任务的下一个任务尚未启动(deleted),则这个尚未启动的任务与其下一要建任务具有相同的栈顶,即当前已建的任务与要建任务栈顶相差 2 单元,而上述尚未启动的任务栈顶与要建任务栈顶相同,当全部任务建完后,满足各任务栈顶相差 2 单元,此 2 单元用于存放其任务入口地址。

4 结 语

KEIL C51 编辑器对变量使用静态分配存储空间,因此存储器管理简化为堆栈管理 。堆栈管理中的栈区移动采取读出压入或弹出写入方式,使栈区移动比较灵活。以上分析可以看出,各个任务所用的堆栈位置是变化的,故 RTX51 Tiny 任务切换过程是动态的,使它可以很容易地运行在没有扩展外部存储器的单片机系统上,且在 51 单片机基础上引入了时间片(time slice)的概念。涉及思想简单,易于理解和移植。但它不支持外部任务切换,限制了任务切换的灵活性。

参考文献

[1] 徐爱钧,彭秀华. 单片机高级语言 C51 Windows 环境编成与应用. 北京:电子工业出版社,2001

[2] KEIL SOFTWARE real - time multitasking executives for the 8051 and MCS 251 Microcontrollers User's Guide

[3] 刘玉宏. KEIL RTX51 TINY 内核的分析与应用. 河海大学:Microcontrollers & Embedded Systems, 2003. 10