μC/OS- II在 AT89C51单片机上的移植

吴鸿霞
(黄石理工学院,湖北 黄石 435003)

摘要:以嵌入式实时操作系统 μC/OS- II的 2. 52版本为基础,详细介绍了 μC/OS- II在 AT89C51单片机的移植过程,讨论了移植过程中应该注意的几个问题,搭建了硬件平台,给出完整的测试过程以验证移植的正确性。

关键词:嵌入式实时操作系统;μC/OS- II;移植
中图分类号: TN361 **文献标识码**: A **文章编号**: 1009-2714(2009)02-0048-04

0 前言

嵌入式技术是目前电子产品设计领域最为热门的技术之一,应用嵌入式操作系统的嵌入式系统设计已成为当前研究的热点,嵌入式系统在工业控制、电子等领域的应用范围不断扩大,嵌入式实时多任务操作系统 (RTOS)可以管理系统中的软硬件资源,是嵌入式软件的运行平台,在嵌入式系统中应用操作系统可以使工程师更专注于应用程序的设计,有利于复杂程序的设计。而操作系统是一个通用的程序,要在自己的嵌入式系统中应用操作系统,必须根据所用 CPU的不同来进行移植。本文将具体论述嵌入式实时多任务操作系统在 AT89C51单片机上的移植和测试。

1 μC/OS- II简介

μC/OS是一种专门为嵌入式设备设计的内核,是 Jean J. Labrosse先生为 68HC11单片机写的[1],是一种免费公开源代码、结构小巧、具有可剥夺内核的 ERTOS。μC/OS- II是 μC/OS的升级版本,其 2. 52版通过了美国航空航天管理 (FAA)认证,可以用于飞机、航天器等与人性命攸关的控制系统中[1,2]。目前已经被移植到 40多种不同结构的 CPU上,运行在从 8位到 64位的各种系统之中。μC/OS- II已经应用在很多领域,比如医疗设备、发动机控制、网络设备、自动提款机及工业机器人等。

2 硬件系统设计

本文以 AT89C51为核心设计了一个简单的硬件平台,介绍说明 μC/OS- II的移植过程,主要实现两个 LED灯自由闪烁功能。AT89C51与发光二极管的连接用 P1. 6和 P3. 7两个端口做 I/O口来控制发光二极管,其连接图如图 1所示。

收稿日期: 2009—02—23
作者简介:吴鸿霞 (1967—),女,湖北武汉人,硕士,副教授.

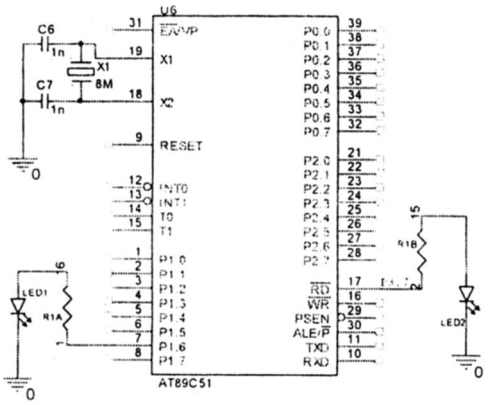

图 1　基于 AT89C51的硬件平台

3　μC/OS - II在 AT89C51上的移植

3.1　移植条件

μC/OS - II的移植必须满足以下条件: 1)由于 μC/OS - II是一个可剥夺型的内核,因此,微控制器的 C编译器必须能够产生可重入型代码; 2)微控制器支持中断,并且能产生定时中断; 3) C编译器支持开关中断; 4)微控制器能支持一定数量的数据存储硬件堆栈[3]。对于 AT89C51及其编译器来说,以上这些条件都是可以满足的。

3.2　移植过程

图 2为 μC/OS - II的结构以及与硬件的关系。从图 2中可以看出 OS_CPU. h、OS_CPU_A. asm、OS_CPU_C. c,这三个文件是与硬件环境相关的,也是移植的关键的部分,因此只要将这部分代码编写好,就可实现系统在硬件的移植。

1) OS_CPU. h文件

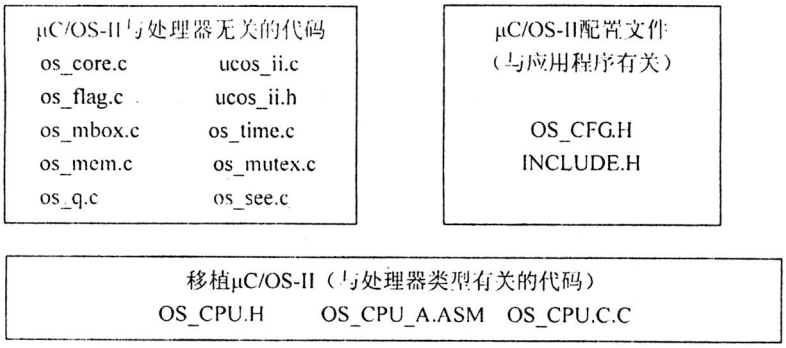

图 2　μC/OS - II的结构以及与硬件的关系

在 OS_CPU. h文件中,包括了用 #define定义的与处理器相关的常数、宏以及类型。该文件由两部分组成,一部分为与编译器相关的的代码,另一部分为与处理器相关的代码。

首先看与编译器相关的部分。因为 AT89C51为 8位单片机系统,采用集成 keil C的编译环境,其处理器为 8位字长[4]。因此我们 OS_STK定义为 8位字长,相应语句为 type def unsigned char OS_STK;而对于 CPU状态寄存器也为 8位,因此应修改,语句为 type def unsigned char OS_CPU_SR. 与处理器相关的部分,在 AT89C51系统中用 C语言代码 EA = 0表示关闭总中断,而用 EA = 1表示开放总中断[5]。因此要将中断部分修改;另外由于 AT89C51单片机的堆栈是从低地址向高地址生长的,因此 OS_STK_GROWTH设为 0,对于堆栈地址从高向低生长的则设为 1。到此,就把 OS_CPU. h文件中要修改的内容就基本上修改完了。

2) OS_CPU_C. c文件

μC/OS - II共定义了 10个函数在该文件中。但是最重要的是 OSTaskStkInit()。其他 9个函数必须声明,但都是对系统内核扩展时用的。OSTaskStkInit()是在用户建立任务时被函数 OSTaskCreate()和 OSTaskCreateExt()调用,是系统内部自己的调用,用来对用户任务的堆栈进行初始化,并使建立好的进入就绪态任务的堆栈与系统发生中断并且将环境变量保存完毕时的栈结构一致。这样就可以用中断返回指令使就绪的任务运行起来。具体的入栈方式要根据不同 MCU而定。需要参考用户使用的 MCU说明书。同时还要考虑 MCU的栈生成方式。这需要根据具体问题来分析,在此不做过多论述。其他 9个函数分别为: void OSInitHookBegin (void); void OSInitHookEnd (void); void OSTaskCreateHook(OS_TCB * ptcb); void OSTaskDelHook(OS_TCB * ptcb); void OSTaskIdleHook (void); void OSTaskStatHook (void); void OSTaskSwHook (void); void OSTCBInitHook (OS_TCB * ptcb); void OSTimeTickHook (void)。

OSTaskStkInit()的示意性代码如下:

```
{OS_STK * stk;
pdat = pdat;
  stk = ptos;
  * stk + + = 15;
```

3) OS_CPU_A. asm文件

在做 μC/OS - II移植时,要求用户编写 4个简单的汇编函数。OSStartHighRdy(); OSCtxSw(); OSIntCtxSw(); OSTickISR()。如果编译器支持插入行汇编代码,就可以将所有与处理器相关的代码放到 OS_CPU_C. c文件中,而不必再有单独的汇编语言文件。

OSStartHighRdy ()函数在多任务系统启动函数 OSStart()中调用。完成的功能是:设置系统运行标志位 OSRunning = TRUE;将就绪表中最高优先级任务的栈指针 Load到 SP中,并强制中断返回。

OSCtxSw()函数是在任务级切换函数中调用的。任务级切换是通过 SWI或者 TRAP人为制造的中断来实现的。ISR的向量地址必须指向 OSCtxSw()。这一中断完成的功能:保存任务的环境变量,将当前 SP存入任务 TCB中,载入就绪最高优先级任务的 SP,恢复就绪最高优先级任务的环境变量,中断返回。这样就完成了任务级的切换。

OSIntCtxSw()在退出中断服务函数 OSIntExit()中调用,实现中断级任务切换。具体完成的任务:调整栈指针,保存当前任务 SP;载入就绪最高优先级任务 SP;恢复就绪最高优先级任务的环境变量;中断返回。

OSTickISR()系统时钟节拍中断服务函,其周期的大小决定了内核所能给应用系统提供的最小时间间隔服务。一般只限于 ms级,该函数具体内容:保存寄存器,OSIntEnter();调用 OSTimeTick();调用 OSIntExit();恢复寄存器;中断返回。

4 应用测试

本测试主要实现的功能为当键按下松开时,灯改变一次状态。第一个任务:第一个灯亮或灭;第二个任务:第二个灯的亮或灭。因为键盘的输入我们要最先响应,所以把键盘输入设为高优先级,设定其优先级为 0,设定第二个任务的优先级为 1。

先编写第一个任务:

```
void mytask1(void) KCREENTRANT
{unsigned char t1 = 0;
InitTimer0();
  for(;;)
  {t1 + +;
  if(LED0 = = 0)
```

```
    {LED0 = 1; }
    OSTimeDly(10); }
}
```

第二个任务代码:

```
void mytask1(void) KCREENTRANT
{unsigned char lastkey = 0xff, temp = 1, t1 = 0;
for(;;)
{t1 ++;
if(LED1 == =1) {LED1 = 0; }
OSTimeDly(10); }
}
```

至此,代码的移植基本完成,下一步的工作将是下载代码,调试。通过编译器移植,代码没有错误,将代码烧录在硬件上。完成后,重新启动单片机。发现两个灯各自闪烁。说明两个任务之间实现了任务的切换,至此我们达到了我们的目的。

5 移植中 keil C51编译器要注意的问题

由于 μC/OS - II是一个可抢占式内核,因此,系统中的绝大多数函数都应该是可重入的。而在 Keil C51编译器中,在函数定义时的默认位都是不可重入的,因此,需要在系统中的每一个函数的声明以及定义处都加上“large reentrant”的修饰符,以保证函数的可重入性。

startup. a51文件是 C51编译器自带的文件,是 C51的初始化代码,单片机复位后先执行这段代码,完成初始化后由它调用 main()。其主要完成定义内部 RAM大小、外部 RAM大小、可重入堆栈位置,以及初始化 AT89C51硬件堆栈指针。因此我们要修改这个文件中的一些变量,比如外部 RAM起始地址变量 XDATA START、是否大模式重入堆栈指针、需初始化标志变量 XBPSTACKTOP等等。程序在软件仿真通过测试后,将其烧录在硬件上,硬件调试也一次成功。

6 结束语

在 μC/OS - II平台下开发程序,首先要掌握内核。通过上述移植过程,能够对任务堆栈、任务调度有深刻理解。自己动手移植,这样有助于更深刻的理解任务的调度过程,而且可以积累一些嵌入式软件调试的经验。总之,在单片机上应用嵌入式操作系统后,由嵌入式操作系统来管理硬件和软件资源,大大方便了应用程序的设计和扩展,也使单片机应用于复杂的场合变得可能。

参考文献:

[1] Jean J Labrosse. 嵌入式实时操作系统 μC/OS - II[M]. 邵贝贝,译. 北京:北京航空航天大学出版社, 2003.

[2]任 哲. 嵌入式实时操作系统 μC/OS - II原理及应用[M]. 北京:北京航空航天大学出版社, 2003.

[3]沈 睿. μC/OS - II在 DSP2407上若干移植问题的研究[J]. 中国水运, 2007, 5(5): 43～44.

[4]马忠梅. 单片机的 C语言应用程序设计[M]. 北京:北京航空航天出版社, 2003.

[5]徐灵伟,刘 飞. 基于 μC/OS - II的嵌入式中断实现与扩展[C]. 北京:中国控制与决策学会, 2007: 999～1002.