文章编号: 1006-2475(2008)04-0017-03

uC/OS_II在 AT89S51单片机上的移植

周 文

(湖南师范大学数学与计算机学院,湖南 长沙 410081)

摘要:详细介绍了 uC/OS_II在 AT89S51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,并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; uC/OS_II
中图分类号: TP316 **文献标识码**: A

Transplantation of uC/OS_II to AT89S51

ZHOU Wen

(School of Mathematics and Computer Science, Hunan Normal University, Changsha 410081, China)

Abstract: This paper researches the method of transplanting open source uC/OS_II to AT89S51, discuses some problem about the transplanting, and gives out the whole testing way to prove the correctness of the transplanting

Key words: embedded real-time operating system; uC/OS_II

0 引 言

目前嵌入式设备已遍布各个应用领域,而大部分的嵌入式设备不需要也没有条件运行操作系统。如,以前许多 MCS51系列单片机组成的小系统就只是利用软件实现简单的控制环路。随着科技的进步和技术的更新,各种微处理器的功能越来越强,各种存储设备的价格变得越来越便宜,为移植操作系统到嵌入式设备上提供了可能。随着所谓后 PC时代的来临,嵌入式系统设计日趋复杂,嵌入式操作系统就必不可少了。本文以在 AT89S51单片机上移植 uC/OS_II为例,阐述了在一般单片机上移植 uC/OS_II的过程[6]。

1 uC/OS_II实时操作系统简介

uC/OS_II是一款源码开放的实时多任务操作系统内核。它具有内核小巧可裁剪、实时性强、支持多任务、可方便移植到多种嵌入式平台等特点,适用于安全性要求苛刻的系统,目前已得到美国联邦航空管理局(federal aviation administration)对用于商用飞机的、符合 RTCA DO-178B标准的认证[5]。uC/OS_II是一个可剥夺式内核,不支持轮转法调度,每个任务都有独立优先级并且使用本地堆栈;在中断退出后并不返回被中断任务,而是直接调度已就绪的高优先级任务执行,从而能够尽快让高优先级的任务得到响应,保证系统的实时性能;同时对临界资源管理借鉴了 Unix中信号量等成熟技术。因此它是一种执行效率高、占用空间小、实时性能优良、可扩展性强的嵌入式实时操作系统[3]。

2 AT89S51单片机

AT89S51单片机是美国 ATMEL公司生产的低功耗、高性能 COMS8位单片机,片内含 4k byte的系统可编程 Flash只读程序存储器,器件采用 ATMEL公司的高密度、非易失性存储技术生产,兼容标准 8051指令系统及引脚。它集 Flash程序存储器及通用 8位微处理器于单片芯片中,既可在线编程(ISP)也可用传统方法进行编程。AT89S51相对于 AT89C51增加了许多新功能,包括: ISP在线编程功能、最高工作频率为 33MHz、具有双工 UART串行通道、内部集成看

收稿日期: 2007-04-16
作者简介:周文(1979-),男,湖南长沙人,湖南师范大学数学与计算机学院硕士研究生,研究方向:嵌入式操作系统。

门狗计时器、双数据指示器、电源关闭标识、全新的加密算法、向下完全兼容 C51系列产品[4]。ATMEL公司的功能强大、低价位的 AT89S51单片机已广泛用于各种控制领域中。

3 移植过程

移植就是使一个实时内核能在其他微处理器或微控制器上运行。基本上就是编写与处理器相关的代码。满足 uC/OS_II移植的处理器的要求如下[1]:

①处理器的 C编译器能产生可重入型代码;

②处理器支持中断,并能产生定时中断;

③用 C语言就可以开/关中断;

④处理器能支持一定数量的数据存储硬件堆栈(可能是几 kB);

⑤处理器有将堆栈指针以及其他 CPU寄存器的内容读出,并存储到堆栈或内存中去的指令。

首先是芯片的中断处理机制,如何开启、屏蔽中断,可否保存前一次中断状态等。其次,芯片是否有软中断或是陷阱指令,又是如何触发的。此外,还需关注系统对于存储器的使用机制,诸如内存的地址空间,堆栈的增长方向,有无批量压栈的指令等[7]。AT89S51单片机满足以上要求,可以完成移植工作。

图 1所示为 uC/OS_II的结构以及与硬件的关系。从图中可以看出完成移植工作主要是将 OS_CPU.H、OS_CPU_A.ASM、OS_CPU_C.C进行修改。

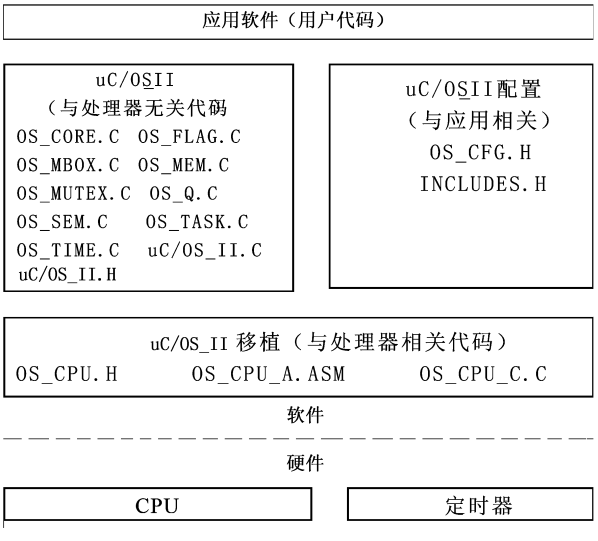

图 1 uC/OS_II硬件/软件体系结构

3.1 堆栈的初始化

uC/OS_II为每个任务建立了任务堆栈,相当于函数调用时保存返回地址和参数一样,用来保存当前任务的状态,保证任务切换能和函数调用一样正确。只不过函数调用时函数堆栈的操作过程是编译器自动完成的,而任务切换时需要模拟一个和编译器类似的任务堆栈的操作过程。堆栈的增长方式有两种,一种是向上增长,一种是向下增长,在 89S51单片机中的堆栈初始指针是指向栈底的,所以应该在 OS_STK_GROWTH中设置为 0。89S51单片机堆栈的结构如图 2。

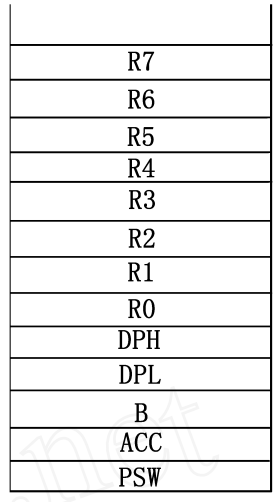

图 2 89S51堆栈结构

3.2 与处理器相关的文件的修改

(1)OS_CPU.H。

OS_CPU.H包括了用 #define语句定义的、与处理器相关的常数、宏以及类型和两个重要的开关中断函数:OS_ENTER_CRITICAL()和 OS_EXIT_CRITICAL()。在 AT89S51单片机中用 EA=0关中断、EA=1开中断来实现。这样定义既减少了程序行数,又避免了退出临界区后关中断造成的死机;uC/OS_II中进行任务切换时通过执行 OS_TASK_SW()来产生中断,要求处理器提供软中断或是陷阱指令来完成这个功能,因为 C51没有软中断指令,所以用程序调用代替:#define OS_TASK_SW() OSCtxSw()。两者的堆栈格式相同,而用 RETI指令复位中断系统。

(2)OS_CPU_C.C。

在 OS_CPU_C.C中定义了 10个函数,其中 9个必须声明,但不一定要包含任何代码,它们是由系统函数调用的接口函数,让用户能在操作系统中加入自己需要的一些功能,如果不需要使用也可以不定义,但要将 OS_CFG.H中的 OS_CPU_HOOKS_EN设为 0.OSTaskCreat()和 OSTaskCreatExt()通过调用 OSTaskStkInit(),初始化任务的栈结构;因此,堆栈看起来就像中断刚发生过一样,所有寄存器都保存在堆栈中。OSTaskStkInit()的示意性代码如下:

```
OS_STK * OSTaskStkInit(void( * task) (void * pd),
void * pdata,
OS_STK * ptos,
INT16U opt);
{
模拟带参数 (pdata)的函数调用;
模拟 ISR向量;
按照预先设计的寄存器值初始化堆栈结构;
返回栈顶指针给调用该函数的函数;
}
```

(3)OS_CPU_A.ASM。

在 OS_CPU_A.ASM中要修改 4个汇编语言函数:

OSStartHighRdy():OSStart()函数调用它来使就绪态任务中优先级最高的任务开始运行。

先将 OSTCBCUR的指针 (指针占 3个字节)存到 DPTR中,再将 OSTCBCur->OSTCBStkPtr的指针存入到 DPTR中,接着用 R5保存堆栈长度,恢复现场堆栈内容,最后恢复堆栈。

OSCtxSw():它是一个任务级的任务切换函数,是在任务需要进行切换时被 OS_TASK_SW()调用。首先获得堆栈的长度和起始地址,然后获得当前 TCB指针,这样就可以获得用户堆栈指针,保存堆栈长度,接着保存仿真堆栈,因为任务切换必须保存当前堆栈内容以免堆栈内容丢失,导致任务恢复失败。最后调用用户程序 LCALL_? OSTaskSwHook,它实现两个功能:OSTCBCur = OSTCBHighRdy, OSPrioCur = OSPrioHighRdy。

OSTickISR():这是时钟节拍中断服务子程序,必须在开始多任务后,即调用 OSStart()后,才可以启动时钟节拍中断,在此它定义了 Tick = 50次/秒 (即 0.02秒/次)。

OSIntCtxSw():它是中断级的任务切换函数,被 OSIntExit()调用,在 ISR中执行中断任务切换。它调整 SP指针去掉在调用 OSIntExit(),OSIntCtxSw()过程中压入堆栈的多余内容。

4　与 Keil C51编译器有关的问题

编译器采用 Keil C51。Keil C51是 Keil公司的一款针对 51系列单片机的编译器,其 uVision2集成开发环境将项目管理、源代码编辑和程序调试等组合在一起,功能强大,使用方便[5]。在 Keil C51环境下移植 uC/OS_II可以直接进行软件仿真,不必下载到硬件上运行。在 Keil C51编译环境中,需要注意几个地方:在 Keil C51中默认编译是不支持可重入代码的,而多任务操作系统需要并发操作有必要使用可重入函数,因此可以在每个 C函数及其声明后面加上 reentrant关键字,将其声明为可重入。pdata、data在 uC/OS_II中用做一些函数的形参,但它同时又是 Keil C51的关键字,会导致编译错误,可以把 pdata改成 ppdata,把 data改成 ddata解决此问题。OSTCBCur、OSTCBHighRdy、OSRunning、OSPrioCur、OSPrioHighRdy这几个变量在汇编程序中用到了,为了使用 Ri访问而不用 DPTR,应该用 Keil C51扩展关键字 DATA将它们定义在内部 RAM中。另外,在 Keil C51中尽量使用指定存储类型的指针 (memory specific pointer)而不使用一般指针 (generic pointer),外部变量都申明为 xdata类型,指向外部变量的指针改为 xdata数据类型的指针,指向函数的指针改为 code数据类型的指针以提高执行效率。在大模式下段名声明的固定格式为:? PR? 函数名? 模块名 SEGMENT CODE。因此需要用上面的格式来声明这 4个函数。C51将所有定义说明的数据标识符转换为大写字符,对函数则根据有无寄存器参数传递和函数是否可重入进行命名。

5　验　证

在修改完以上 3个文件后,uC/OS_II移植工作就完成了,下面进行移植工作测试[1]:

①确保编译器、汇编编译器及链接器正常工作。这时可给出一个最简单的主函数,该函数没包含任何代码,只是为了验证是否可以编译出正确的代码。

②接下来验证 OSTaskStkInit()和 OSStartHighRdy()函数。将代码下载到调试器,单步执行 main()函数直到调用 OSStartHighRdy()。这时编译器会切换到汇编语言模式下。OSStartHighRdy()会开始运行第一个任务,此时无其他任务只有 OS_TaskIdle()。继续单步执行,如果调试器在 OS_TaskIdle()的循环中运行,且在无限循环中执行过几次,则验证成功。

③验证 OSCtxSw()函数。在这里添加一个应用程序,并不断切换到空闲任务。在此之前要正确设置软中断或指令陷阱 TRAP,使之指向 OSCtxSw()函数。单步执行 OSTimeDly()函数,接着调用 OS_Sched(),而 OS_Sched()调用汇编语言编写的 OSCtxSw()函数。如果单步进入 OSCtxSw()代码,看到任务寄存器已保存至它的堆栈而 OS_TaskIdle()的寄存器正被调入 CPU,且从中断返回时在 OS_TaskIdle()的任务中,则验证函数正确。　　(下转第 22页)

(△ρ, △θ, △r),在每小单元设一累加器,然后对图像中每一有效点进行变换。待图像中全部有效点都变换完成后,对各小单元进行检测。找出公共点,并将其代入方程:

$$r_i^2 = (x - \rho_i \cos\theta_i)^2 + (y - \rho_i \sin\theta_i)^2$$

从而得出逼近的圆的方程。

在以上检测中,若在落入次数最多的小单元附近同时存在另外一个落入次数比较多达到临界值的其他几个小单元(由划分小单元的精度决定,这种情况经常存在),此时可把这些单元忽略,认为它们对应着图像域中同一直线或圆。当落入次数达到临界值的单元彼此距离较远时,则可认为它们对应着图像域中不同的直线或圆。

由于检测圆的参数有三个,目前暂时无法得到它的 Hough变换平面图。但仍可以通过计算机采用以上方法编程来实现。不管图像中有多少个圆,只要它们之间的距离满足要求,则可用跟多条直线检测一样的方法检测出来。图 3中有五条直线,则其 Hough变换图中亦有五个亮点,若图中有多圆,则其变换域中同样也有同样个数的公共点。然后应用这些公共点参数反变换即可得到图中的逼近圆。

4 结束语

本文研究了 Hough变换的性质及其在直线检测中的应用,并将其推广应用于圆的检测。结果表明,Hough变换在几何特征检测方面有着独特的性能,将待检测目标从目标空间转换到参数空间,避免了在目标空间检测时的目标分类、目标编码等复杂运算,使得被测参数的测量变得简单易行。

在车牌自动识别系统中,对倾斜的车牌进行校正时便可采用 Hough变换。由于规则的车牌边缘是直线,故可采用 Hough变换求出其边缘的倾角,再进行相应的旋转校正。圆的检测在图像形态识别领域中占有着很重要的地位。根据圆孔或圆弧的图像求取其中心坐标和半径等特征参数是计算机视觉领域中一个很重要的问题。而 Hough变换在图像噪声上表现出来的鲁棒性使其成为图像轮廓检测中的常用的方法。

参考文献:

[1] 阮秋琦. 数字图像处理学 [M]. 北京:电子工业出版社, 2001.

[2] 赵书安,冯少彤,聂守平. Hough变换在几何特征检测中的应用 [J]. 南京师范大学学报 (工程技术版), 2006, 6 (4): 66-70.

[3] 何东健. 数字图像处理 [M]. 西安:西安电子科技大学出版社, 2003.

[4] Kenneth R Castleman. Digital Image Processing[M]. Englewood Cliffs: Prentice Hall, 1996: 460-468.

[5] 何东健,耿楠,张义宽. 数字图像处理 [M]. 西安:西安电子科技大学出版社, 2004.

[6] 王强,胡维平,等. Hough变换实时检测算法研究 [J]. 计算机工程与设计, 2001, 22 (3): 76-80.

[7] 林金龙,石青云. 用点 Hough变换实现圆检测的方法 [J]. 计算机工程, 2003, 29 (11): 17-18.

[8] 高希报. 图像中几种实用的目标定位方法研究与应用 [D]. 南京:南京理工大学硕士学位论文, 2005.

(上接第 19页)

④验证 OSIntCtxSw()、OSTickISR()函数。当进入测试任务时,初始化时钟中断源,调用 OSTimeDly(),使 OSCtxSw()将任务切换到空闲任务。当时钟节拍中断时调用 OSTickISR(),继而调用 OSTimeTick()。OSTimeTick()将任务的 OSTCBDly计数器递减到 0,使任务就绪。而当 OSTickISR()完成并调用 OSIntExit()时, ISR就不会返回空闲任务,而是做任务切换,回到测试任务。至此,移植的 uC/OS_II能够正常工作,可以添加应用任务了。

6 结束语

uC/OS_II作为一个优秀的实时操作系统已经被移植到各种体系结构的微处理器上,而 51单片机在嵌入式领域的应用十分广阔。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,简化了应用程序的设计,且使得应用系统功能更加完善,因此具有十分重要的意义。

参考文献:

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

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

[3] 任哲. 嵌入式实时操作系统 uC/OS_II原理及应用 [M]. 北京:北京航空航天大学出版社, 2003.

[4] 吴金戎,等. 8051单片机实践与应用 [M]. 北京:清华大学出版社, 2002.

[5] 罗蕾. 嵌入式实时操作系统及应用开发 [M]. 北京:北京航空航天大学出版社, 2005.

[6] 陈章龙,等. 嵌入式技术与系统 —— Intel XScale结构与开发 [M]. 北京:北京航空航天大学出版社, 2004.

[7] [美 鲍尔. 嵌入式微处理器系统设计实例 [M]. 苏建平,等译. 北京:电子工业出版社, 2004.