基于DSP56F807的μC/OS－II移植和实现*

韩冻冰[1],周　缘[2]
(1.盐城工学院 优集学院,江苏 盐城　224003;2.上海大学 通信与信息工程学院,上海　200444)

摘　要:首先分析实时操作系统μC/OS－II的系统结构和DSP56F807硬件平台特性,其次编写与硬件相关的代码,将μC/OS－II实时内核移植到DSP56F807上,并进行移植后的测试,最后提出一些在μC/OS－II移植过程中的问题。

关键词:μC/OS－II;DSP;实时操作系统;移植
中图分类号:TP274.2　　**文献标识码**:A　　**文章编号**:1671－5322(2007)02－0032－04

嵌入式系统设计已经改变和影响着我们的生活,时至今日,其带来的工业产值已超过了1万亿美元,研究嵌入式系统的开发成为计算机和电子技术工业应用的热点,要想实现嵌入式的应用,最基础的工作就是实现嵌入式操作系统在相关处理器平台的移植。

1　μC/OS－II系统结构

μC/OS是由Jean J. Labrosse在使用商业软件后,自己研究开发的实时内核,经过10多年的发展,μC/OS升级到μC/OS－II[1]。

基于ANSIC编写的μC/OS－II的源码具有很强的移植性,其面向中小型嵌入式系统,可以在大多数8位、16位和32位的处理器上运行,与微处理器硬件相关部分则采用与之相应的汇编语言编写的,解决底层设计的需要。

图1说明了μC/OS－II的软硬件体系结构。应用程序处于整个系统的顶层,每个任务都可以认为自己独占了CPU,因而可以设计成为一个无限循环。与处理器无关的代码提供了μC/OS－II的系统服务,应用程序使用这些API函数进内存管理、任务间通信以创建、时间管理方面的操作。

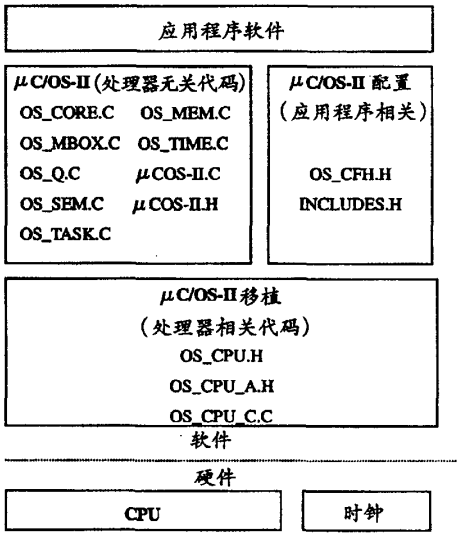

图1　μC/OS－II硬件/软件体系结构
Fig. 1　The architecture of the hardware/software of μC/OS－II

2　DSP56F807体系结构

DSP56F807是Motorola公司于2001年推出的低价位系列之一的16位数字信号处理器产品,它是基于内核结构DSP56800家族中的一员。DSP56F807将DSP和MCU集成在一起,兼有DSP强大的运算能力和微控制器丰富灵活的片上外设模块,并且价格低廉,外设模块配置灵活,为嵌入式应用开发提供了低成本高效率的解决方案[2]。

* 收稿日期:2007－03－11
作者简介:韩冻冰(1980－),男,江苏盐城市人,硕士,助教,主要研究方向为嵌入式系统应用。

DSP56800内核采用先进的哈佛结构,将存储器划分为数据区和程序区两个空间,并行CPU集成三级流水作业的三个同时执行单元:算术逻辑ALU、地址产生单元AGU、程序控制单元PCU,每个指令周期允许6次运算操作,具有高度并行特点。微处理器风格的编程模式以及灵活的寻址模式和位操作指令,使用户在写指令时不必担心DSP的复杂性。优化的指令集可直接产生高效的、精简的代码,对C/C++编译器来说同样也是高效的。

3　移植条件

μC/OS-II是一个实时内核,支持多任务、中断嵌套、信号量、消息邮箱和内存管理功能,所以编译后大约在6~10 kB,即使保留最核心的代码,也要有2 kB大小。此外,μC/OS-II为每个任务开辟一个任务堆栈,这也要占用较大的RAM空间,所以移植一个实时操作系统对处理器有硬件要求[3]。

(1)处理器的C编译器能产生可重入代码

可重入的代码指的是一段代码能被多个任务同时调用,这段数据不会被破坏。也就是说,可重入型函数在中断的时候被其他的任务重新调用,而函数中的数据不会受到影响,能够重复使用。

(2)在程序中可以利用C语言打开或者关闭中断

在μC/OS-II中,通过处理器支持的宏OS_ENTER_CRITICAL()和OS_EXIT_CRITICAL()来关闭或打开系统中断,在DSP56F807处理器上可以设置相应的寄存器来关闭或者打开系统的所有中断。

(3)处理器支持中断,并且能产生定时中断

μC/OS-II是通过处理器产生的定时器中断来实现多任务之间的调度。DSP56F807处理器可以产生定时器中断。

(4)处理器能够支持一定数量的数据存储硬件堆栈

(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的指令

寄存器入栈和出栈是μC/OS-II多任务调度的基础,DSP56F807处理器中有专门的指令处理堆栈,在μC/OS-II进行任务调度时,会把当前任务的CPU寄存器存放到此任务的堆栈中,再从另一个任务的堆栈中恢复原来的工作寄存器,继续运行另一个任务。

DSP56F807开发板的数字信号处理器属于典型应用的DSP56800系列产品,采用8 MHz外部晶振,利用内部压控振荡器和锁相环产生80 MHz总线时钟,支持定时中断,具有60 kB程序Flash区和2 kB程序RAM区,以及8 kB的数据Flash区和4 kB的数据RAM,另外还可以分别外扩64 kB数据和程序存储区。本次移植采用的与DSP56F807配套的交叉编译器CodeWarrior for DSP56F807 V5.1可以产生可重入代码,同时支持在C程序中嵌入汇编语句。所以,开发者具备移植的条件,能够把μC/OS-II移植到DSP56F807上。

4　μC/OS-II实时内核移植

根据上述的μC/OS-II的体系结构以及与硬件的关系,移植工作主要是改写OS_CPU.H、OS_CPU.C、OSCPUA.ASM3个文件,而基于μC/OS-II编写应用程序时,需要改写OS_CFG.H和INCLUDES.H两个文件。OS_CFG.H文件中预定义了很多配置内核服务功能的开关量以及一些与μC/OS-II相关的常量,用户可以根据具体应用程序的特点,配置该文件。

4.1　改写INCLUDES.H

INCLUDES.H是主头文件,在一个工程中所有后缀为.C的文件开始部分都包括INCLUDES.H。根据不同的处理器和面对不同的应用程序,该文件可以被改写,相应的增加自己的头文件。

```
/* SDK 定义的头文件 */
#include "port.h"
#include "mempx.h"
/* 与处理器相关的头文件 */
#include "os_cpu.h"
/* 应用配置头文件 */
#include "os_cfg.h"
/* 与处理器无关的系统头文件 */
#include "ucos_ii.h"
```

如果用户想在应用程序中添加自己的头文件,可以修改或者添加自己的头文件。

4.2　改写OS_CPU.H

下面是针对DSP56F807编写的OS_CPU.H文件内容,其中包括与处理器相关的常量、宏和数据类型。

```
typedef unsigned int OS_STK;
```

```
typedef unsigned short OS_CPU_SR;
typedef unsigned int OS_EVENT;
#define INT8S signed char
#define INT8U unsigned char
#define INT16S signed int
#define INT16U unsigned int
#define INT32S signed long
#define INT32U unsigned long
#define OS_ENTER_CRITICAL( ) asm sei
#define OS_EXIT_CRITICAL( ) asm cli
# define OS _ STK _ GROWTH 0 /* 定义DPS568xx 堆栈由低地址向高地址 */
#define OS_TASK_SW( ) asm(swi) /* 软中断 */
```

μC/OS-II在移植过程中需要定义数据类型来满足处理器字长的要求。对于不同的处理器而言,数据入堆栈时堆栈指针的增长方向不一样,所以要设定堆栈的增长方向。μC/OS-II定义两个宏OS_ENTER_CRITICAL( )和OS_EXIT_CRITICAL( )保护临界区,这样可以先禁止中断再访问程序代码的临界区,并且在访问完毕后重新允许中断,从而保护临界区代码免受多任务或中断服务程序的破坏。

4.3 编写5个汇编程序OS_CPU_A.ASM

这5个与Motorola处理器底层紧密相关的汇编程序是μC/OS-II移植工作的重点,主要是通过调用C语言相对应的程序段,CodeWarrior编译器支持插入汇编语言的代码,用户可以将所有与处理器相关的代码放到OS_CPU_C.C文件中。

(1)OSTaskStkInt( )用来初始化任务的堆栈,将所有的寄存器保存到堆栈中的情形一样。当前任务初始化完成后,OSTaskStkInt( )返回新的堆栈指针stk,在OSTaskCreateHook( )执行时将会调用OSTaskStkInt( )的初始化过程,然后通过OSTCBInit( )函数调用将返回的SP指针保存到该任务的TCB块中。

(2)OSStartHighRdy( )运行优先级最高的就绪任务,程序将所有寄存器从函数OSTaskStkInit( )创建的任务堆栈中恢复出来,并且执行中断返回。OSStartHighRdy( )必须调用OSTaskSwHook( ),但OSTaskSwHook( )函数可以是个空函数,只有把OS_CFG.H的OS_CPU_HOOKS_EN设为1,OSTaskSwHook( )的代码才能生效。另外,OSTaskHighRdy( )还必须在最高优先级任务运行之前后设置OSRunning为TRUE。

(3)OSCtxSw( )任务级的调度,任务遇到阻塞请求CPU调度时执行该函数,和OSIntCtxSw( )不同,该任务函数将被挂起的任务寄存器推入堆栈,然后将最高优先级的任务处理器寄存器从任务堆栈中恢复出来,使之继续执行。

(4)OSIntCtxSw( )中断级任务调度,目的是将被中断的任务参数保存到其堆栈中,然后由CmnCtxSw完成被中断任务堆栈指针的保存、高优先级任务执行环境的恢复与执行,与OSCtxSw( )相比,OSIntCtxSw( )只要在OSCtxSw( )代码前增加调整SP指令。

(5)OSTickISR( )需要用户提供一个时钟资源来实现时间的延时和期满功能。DSP56F807有专门的硬件定时器产生,定时中断由SDK中的CONFIG.H设置为每1 ms产生一次中断,即定时节拍的频率为1 000 Hz。

4.4 C语言编写的OS_CPU_C.C

这些函数必须声明,但不一定要包含代码,因为这些函数是提供给用户扩展μC/OS-II的功能用的。在OS_CFG.H中的OS_CPU_HOOKS_EN设置为1时才产生这些函数的代码。该文件包含如下几种函数OSTaskCreateHook( )、OSTaskDelHook( )、OSTaskSwHook( )、OSTaskIdleHook( )、OSTaskStatHook( )、OSTimeTickHook( )、OSInitHookBegin( )、OSInitHookEnd( )、OSTCBInitHook( )。

5 测试μC/OS-II移植代码

做完DSP56F807μC/OS-II移植后,紧接着就是验证移植的μC/OS-II是否正常工作,而这是移植中最复杂的一步。在Metrowerks公司的CodeWarrior编译器下进行所移植后的,基于硬件DSP56F807平台的软件测试[4],以验证移植后的软件和硬件是完全交互的。

为了证明μC/OS-II移植的有效性,要使用某些系统服务函数。我们不可能把所有的服务都允许了,这样要进行很繁琐的操作,而且要占有相当的资源,根据OS_CFG.H的配置文件,进行内核服务函数的使能。

测试任务的实例主要完成统计DSP56F807的中央处理器的利用率。在本测试实例中,系统执行15个任务,其中μC/OS-II有自己2个任务:空闲任务和统计任务。其余的13个任务中,

任务 TaskStart( )是在主函数 OSTaskCreate( )中建立,另外的 12 个任务函数在任务 TaskStart( )中建立,这 12 个任务都是基于相同的代码 Task( ),每个函数执行 10000 次的信号量请求与释放操作,测试结果图 2 所示:

该实例运行结果表明,系统运行这 9 个任务的过程,DSP56F807 的 CPU 利用率为 2%。经过测试,本次移植工程中的关键,OS_CPU_A. ASM 中 5 个函数(表 1)改写后运行稳定且正确。

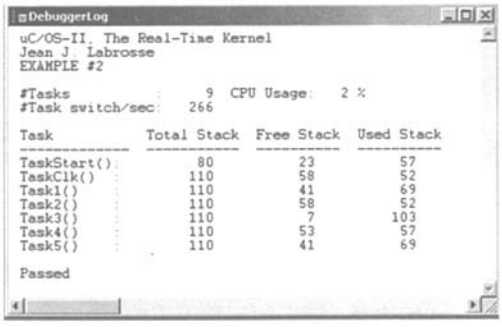

图 2　移植测试结果

Fig. 2　The result of testing the porting

表 1　OS_CPU_A. ASM 任务函数

Table 1　The functions of OS_CPU_A. ASM

函数	函数调用名	函数功能
任务堆栈初始化函数	OSTaskStkInit( )	初始化任务的堆栈
最高优先级就绪任务启动函数	OSStartHighRdy( )	运行优先级最高的就绪任务
任务级任务切换函数	OSCtxSw( )	通过一次软中断来实现任务的切换
中断级任务切换	OSIntCtxSw( )	在中断服务程序中完成任务的切换
时钟节拍中断服务	OSTickISR( )	提供时钟中断服务

6　结束语

μC/OS - II 仅是一个实时多任务的内核,移植了 μC/OS - II 以后,离实际的应用还是有一段距离的。就 μC/OS - II 内核通信机制来说,还有许多需要完善和修改的地方,要想实现一个相对完整、实用的嵌入式实时多任务操作系统(RTOS),还需要做相当多的扩展性工作。

参考文献:

[1] 邵贝贝,许庆丰,王若鹏. 一个源码公开的实时内核[J]. 单片机与嵌入式系统应用,2001(9):70 - 75.

[2] 邵贝贝,龚光华,薛涛,等. Motorola DSP 型 16 位单片机原理与实践[M]. 北京:北京航空航天大学出版社,2003.

[3] Jean J, Labrosse. μC/OS - II — μC/OS - II The Real - Time Kernel Second Edition[M]. 邵贝贝译. 第 2 版. 北京:北京航空航天大学出版社,2003.

[4] 邓世伟. 嵌入式软件的测试方法和工具[J]. 单片机与嵌入式系统应用,2001(4):26 - 28.

The Porting and Implementation of μC/OS - II Based on DSP56F807

HAN Dong - bing[1], ZHOU Yuan[2]

(1. UG School of Yancheng Institute of Technology, Jiangsu Yancheng 224003, China;
2. School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China)

Abstract: This paper has analyzed the RTOS μC/OS - II kernel and DSP56F807's principle and performance, then ported it to DSP56F807, compiled the hardware - associated code and tested the result. Finally it has brought forth some questions during the porting.

Keywords: μC/OS - II; DSP; porting; RTOS