

基于 ARM 的嵌入式操作系统 $\mu\text{C}/\text{OS}-\text{II}$ 的移植研究

邓中亮, 何双亮

(北京邮电大学 电子工程学院, 北京 100876)

摘要:随着 ARM 技术的广泛应用, 建立基于 ARM 构架的嵌入式操作系统已成为当前研究的热点。文中结合实例论述了基于 ARM 内核的微处理器上的嵌入式 $\mu\text{C}/\text{OS}-\text{II}$ 操作系统的移植技术, 介绍了 $\mu\text{C}/\text{OS}-\text{II}$ 系统主要特点, 给出了移植条件及移植的实现过程, 同时对编写启动代码进行了说明并测试验证通过。

关键词:ARM; 嵌入式操作系统; 移植; $\mu\text{C}/\text{OS}-\text{II}$

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2007)10-0004-03

Research of Porting Embedded $\mu\text{C}/\text{OS}-\text{II}$ Operating System Based on ARM

DENG Zhong-liang, HE Shuang-liang

(School of Electronic & Engineering, Beijing Univ. of Post & Telecom., Beijing 100876, China)

Abstract: With the applications of ARM technique, building the embedded operating system based on ARM processor has been a hot point of research. Discusses the porting of $\mu\text{C}/\text{OS}-\text{II}$ RTOS on ARM7 kernel architecture. Introduce primary characteristics of $\mu\text{C}/\text{OS}-\text{II}$ system and presents the prerequisite and realization process of transplant. Finally, explain the reason of program start code and validate through testing.

Key words: ARM; embedded operating system; porting; $\mu\text{C}/\text{OS}-\text{II}$

0 引言

嵌入式应用中移植微型的操作系统, 一方面能够把整个系统要实现的功能进行分解, 并用不同的任务来实现这些应用功能, 使得应用系统的软件结构模块化。一方面, 嵌入式系统由于以多任务管理作为基础, 可以满足应用系统实时性的要求, 有助于提高整个系统的稳定性和可靠性。不同的嵌入式应用其软硬件结构和算法各有特点, 因而嵌入式系统的移植必须是对具体的应用专门设计的。 $\mu\text{C}/\text{OS}-\text{II}$ 是一种简单、高效、源代码公开的实时嵌入式操作系统, 被广泛应用到各种嵌入式系统中。鉴于 $\mu\text{C}/\text{OS}-\text{II}$ 良好的扩展性和可移植性, 因此将其移植到嵌入式处理器上, 对于提高产品的质量, 减少开发周期和降低成本方面有着重要的意义。

$\mu\text{C}/\text{OS}-\text{II}$ ^[1] 是源码公开基于 $\mu\text{C}/\text{OS}$ 的嵌入式实时操作系统内核, 支持 x86, ARM, PowerPC, MIPS 等众多体系结构。 $\mu\text{C}/\text{OS}-\text{II}$ 是一个完整的、可移植、固

化及裁剪的占先式实时多任务内核, 是基于优先级的, 即总是让就绪状态中优先级高的先执行, 因此实时性比非占先式的内核要好。它包含了实时内核、任务管理、任务间通信同步(信号量、邮箱、消息队列等)和内存管理等功能; 它的绝大部分代码是用 ANSI C 语言编写的, 结构简单可移植性强。目前, $\mu\text{C}/\text{OS}-\text{II}$ 可以在绝大多数 8 位、16 位、32 位、64 位微处理器(MPU)、微控制器(MCU)和数字信号处理器(DSP)上运行, 被广泛应用于网络设备、医疗设备、发动机控制、自动提款机及工艺机器人等众多领域。

1 移植平台介绍

文中讨论的嵌入式目标平台采用的是凤凰微电子有限公司自主开发的基于 32 位 ARM7TDMI 核的 SSC 嵌入式 RISC 处理器。具有 32kSRAM、8M 的片内 Flash 和 32M 的片外 Flash, 主频高达 50MHz。支持 SRAM、SDRAM 和 FLASH 等多样大容量外部存储器。文中所用平台在系统的统一编址空间里, 经重映射后的地址空间分配为: SRAM 位于 0x40000000 ~ 0x4000xxxx, 片内 P-FLASH 位于 0x00000000 ~ 0x0003FFFF, 片内 D-FLASH 位于 0x20000000 ~

收稿日期: 2007-01-07

作者简介: 邓中亮(1965-), 男, 教授, 博士生导师, 研究方向为智能通信和嵌入式系统研究。

0x2001FFFF, 片外 D_FLASH 位于 0x80000000 ~ 0x80xxxxxx。要保证 $\mu\text{C}/\text{OS}-\text{II}$ 移植到微处理器后能正确运行, 处理器需具备如下特性^[2]:

1) 处理器的 C 编译器支持可重入函数。

2) 在程序中可以打开和关闭中断。

3) 处理器支持中断, 并且能产生定时器中断, $\mu\text{C}/\text{OS}-\text{II}$ 是通过定时器中断来实现多任务的调度, 即时间片的产生。 $\mu\text{C}/\text{OS}-\text{II}$ 是通过处理器产生的定时器的中断来实现多任务之间调度的。

4) 处理器要具有一定的硬件堆栈数量。

5) 处理器要有将堆栈指针和其他 CPU 寄存器存储和读出堆栈(或者内存)的指令。

基于 ARM7TDMI 核^[3]的 SSC 嵌入式处理器完全满足上述要求。

2 $\mu\text{C}/\text{OS}-\text{II}$ 的移植

2.1 $\mu\text{C}/\text{OS}-\text{II}$ 内核移植要点

$\mu\text{C}/\text{OS}-\text{II}$ 其 90% 的代码是用 C 语言写的, 可以直接移植到有 C 语言编译器的处理器上, 但是移植工作主要都集中在多任务切换的实现上, 因为这部分代码用来保存和恢复 CPU 现场(即写/读相关寄存器)不能用 C 语言, 只能使用汇编语言完成。将 $\mu\text{C}/\text{OS}-\text{II}$ 移植到 ARM 处理器上, 需要修改三个与 ARM 体系结构相关的文件。以下介绍这三个文件的移植工作^[4]。

(1) OS_CPU.H 文件。

* 数据类型定义: 数据类型的修改与所用的编译器相关, 不同的编译器使用不同的字节长度表示同一数据类型。

```
typedef unsigned char BOOLEAN;
```

```
typedef unsigned char INT8U;
```

```
typedef signed char INT8S;
```

```
typedef unsigned int INT16U;
```

```
typedef signed int INT16S;
```

```
typedef unsigned long INT32U;
```

```
typedef signed long INT32S;
```

```
typedef float FP32;
```

```
typedef double FP64;
```

* 堆栈单位: 在任务切换时, CPU 现场的寄存器将保存在当前运行任务的堆栈中, 所以 OS_STK 数据类型应该与 CPU 的寄存器长度一致。typedef unsigned int OS_STK;

* 堆栈增长方向: SSC 堆栈由高地址向低地址增长。在函数调用时, 入口参数和返回地址一般保存在当前任务的堆栈中, 编译器的编译选项和由此生成的堆栈指令就会决定堆栈的增长方向。# define OS_STK_GROWTH;

* 宏定义: 包括开关中断的宏定义, 以及进行任务切换的宏定义。

```
# define OS_ENTER_CRITICAL() ARMDisable Int()
```

```
# define OS_EXIT_CRITICAL() ARMEnable Int()
```

```
# define OS_TASK_SW() OSCtxSw()
```

(2) OS_CPU.C 文件。

* 任务堆栈初始化: 任务初始化时的堆栈设计^[5], 也就是在堆栈增长方向上如何定义每个需要保存的寄存器位置, 在 ARM 体系结构下, 任务堆栈空间由高至低依次将保存着 PC, LR, R12, CPSR, SPSR 等寄存器, 如图 1 所示。移植中有两点需要说明: 一是当前任务堆栈初始化完成后, OSTaskStkInit() 返回新的堆栈指 STK, OSTaskCreate() 执行时, 将会调用 OSTaskStkInit() 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; 二是初始状态的堆栈是模拟了一次中断后的堆栈结构, 因为任务创建后并不是直接就获得执行, 而是通过 OSSched() 函数进行调度分配, 满足执行条件后才能获得执行。为了使这个调度简单一致, 就预先将该任务的 PC 指针和返回地址 LR 都指向函数入口, 以便被调度时从堆栈中恢复刚开始运行时的 CPU 现场。

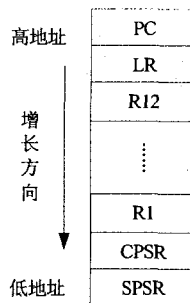


图 1 任务堆栈寄存器

* 系统钩子函数: 移植中需实现几个操作系统规定的 hook 函数, 如下:

```
OSSTaskCreateHook()
```

```
OSTaskDelHook()
```

```
OSTaskSwHook()
```

```
OSTaskStatHook()
```

```
OSTimeTickHook()
```

若无特殊需求, 只需简单地将它们都实现为空函数即可。

(3) OS_CPU.A.S 文件。

* OSStartHighRdy(): 此函数是在 OSStart() 多任务启动后, 负责从最高优先级任务的 TCB 控制块中获得该任务的堆栈指针 SP, 通过 SP 依次将 CPU 现场恢复, 这时系统就将控制权交给用户创建的该任务进程, 直到该任务被阻塞或者被其他更高优先级的任务抢占 CPU。该函数仅在多任务启动时被执行一次, 即执行

最高优先级任务,之后多任务的调度和切换由以下函数实现。

* OSCtxSw():任务级的上下文切换,当任务因为被阻塞而主动请求 CPU 调度时被执行,由于此时的任务切换在非异常模式下进行,因此区别于中断级别的任务切换。它的工作是先将当前任务的 CPU 现场保存到该任务堆栈中,然后获得最高优先级任务的堆栈指针,从该堆栈中恢复此任务的 CPU 现场,使之继续执行。这样就完成了一次任务切换。

* OSIntCtxSw():中断级的任务切换,在时钟中断 ISR(中断服务例程)中发现有高优先级任务等待的时钟信号到来,则在中断退出后并不返回被中断任务,而是直接调度就绪的高优先级任务执行,从而能够尽快地让高优先级的任务得到响应,保证系统的实时性能。其原理基本上与任务级的切换相同,但是由于进入中断时已经保存了被中断任务的 CPU 现场,因此不再进行类似的操作,只需对堆栈指针做相应调整。

* OSTickISR():时钟中断处理函数,其主要任务是负责处理时钟中断,调用系统实现的 OSTimeTick 函数,如果有等待时钟信号的高优先级任务,则需要在中断级别上调度其执行。其他相关的两个函数是 OSIntEnter()和 OSIntExit(),都需要在 ISR 中执行。

* OS_ENTER_CRITICAL() & OS_EXIT_CRITICAL():这两个函数分别是退出临界区和进入临界区的宏指令实现。主要用于在进入临界区之前关闭中断,在退出临界区的时候恢复原来的中断状态。它的实现比较简单,可以直接开关中断来实现,也可以通过保存关闭/恢复中断屏蔽位来实现。

2.2 SSC 快速自启动程序设计

为保证移植代码以及应用代码在 SSC 处理器上能够执行,还必须为处理器编写启动代码^[6]。启动代码简单地说就是为 Main()函数的运行准备环境。包括初始化异常向量表、堆栈以及与目标平台的相关定义。启动程序用汇编语言编写,要完成的任务包括:硬件初始化,系统存储系统的配置,复制二级中断向量表等(如图 2 所示)。由于系统的资源有限,程序首先固化在 ROM 中运行。为提高系统的实时性,加快代码的执行速度,系统启动后程序被搬移到 RAM 中,因为 RAM 的存取速度要比 ROM 快得多,这样大大提升系统的性能。

3 系统测试

移植完成后的系统包括 4 个部分:

1)自启动程序;

2) μ C/OS-II 系统文件;

3)移植代码;

4)应用程序。

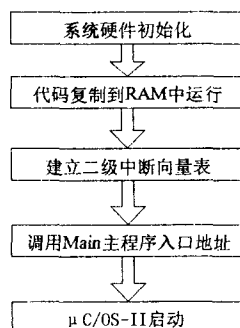


图 2 系统启动

开发环境是 ARM 公司提供的 ADS1.2 (ARM Developer Suite)。为了验证 μ C/OS-II 移植的成功,创建了下面 3 个任务:

(1)D/A 输出;

(2)A/D 采集;

(3)LCD 显示。

使用信号量来实现模拟量的采集(A/D)和模拟量的控制(D/A)之间的通讯,同时模拟量和控制量通过 LCD 显示系统运行结果。可以观测到,LCD 显示的数据随模拟量的输出而变化,从而验证移植成功。

4 结论

μ C/OS-II 具有很强的实时性,已经被移植到各体系结构的微处理器上。着重分析了将 μ C/OS-II 移植到 ARM 构建的嵌入式系统上的主要技术和基本流程,掌握这些移植的技术和流程,对于开发嵌入式系统是十分重要的。目前已经成功移植 μ C/OS-II 到 SSC 目标平台,并能顺利启动和稳定运行,达到实时性要求。

参考文献:

- [1] Jean L. 嵌入式实时操作系统 μ C/OS-II[M]. 邵贝贝译. 北京:北京航空航天大学出版社,2003.
- [2] 王田苗. 嵌入式系统设计与实例开发[M]. 北京:清华大学出版社,2002.
- [3] ARM Architecture Reference Manual[M]. [s.l.]: Advanced RISC Machines Ltd,2000.
- [4] 周立功. ARM 嵌入式系统实验教程[M]. 北京:北京航空航天大学出版社,2004.
- [5] 马忠梅. ARM 嵌入式处理器结构与应用基础[M]. 北京:北京航空航天大学出版社,2002.
- [6] 杜春雷. ARM 体系结构与编程[M]. 北京:清华大学出版社,2004.