文章编号:1006-2475(2007)01-0031-02

实时操作系统 μC/OS-Ⅱ在TMS320VC5509上的移植

鲍 劼[1],张 蓓[2]

(1.中国矿业大学图书馆系统部,江苏 徐州 221008;2.中国矿业大学计算机学院,江苏 徐州 221008)

摘要:μC/OS-Ⅱ是一种基于优先级占先式的嵌入式操作系统,提供了任务管理、时间管理、内存管理以及任务之间的通信和同步等功能。由于其可移植性和可裁剪性,以及成功地应用到了商业系统中,本文介绍了该系统在TMS320VC5509 DSP上面的移植,并且通过仿真证明了其正确性。

关键词:嵌入式系统;μC/OS-Ⅱ TMS320VC5509;移植

中图分类号:TP316 文献标识码:A

Transplantation of Real-time Operating System μC/OS-Ⅱ on TMS320VC5509

BAO Jie[1], ZHANG Bei[2]

(1. Library of China University of Mining and Technology, Xuzhou 221008, China;
2. School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221008, China)

Abstract: μC/OS-Ⅱ is a typical real-time operation system, which is based on the priority-based scheduling mechanism, providing task scheduling, memory management, the communication and synchronization between tasks. It has been successfully applied in hundreds of commercial embedded system. This paper introduces the adapting method to transplant this OS in TMS320VC5509, which has been proved the validity with emulator.

Key words: embedded system; μC/OS-Ⅱ TMS320VC5509; transplantation

0 引 言

目前嵌入式系统在工业控制、家电、移动电话、数码相机等各种领域的应用越来越广泛。由于嵌入式操作系统的引入,使整个嵌入式系统开发层次化、模块化。通过这种层次式的设计方式,缩短了开发的时间,提高了开发的效率,而且减轻了开发人员的负担。

μC/OS-Ⅱ是一个源代码开放的多任务实时操作系统,提供了任务管理、时间管理、内存管理以及资源管理四大功能,并且可移植性和可裁剪性很强。

本文介绍了μC/OS-Ⅱ在TMS320VC5509 DSP上面的移植,通过测试程序证明了移植的成功。

1 μC/OS-Ⅱ操作系统在TMS320VC5509 DSP上面的移植

μC/OS-Ⅱ与硬件相关的函数有以下5个:OSTaskStkInit( ),OSStartHighRdy( ),OSCtxSw( ),OSTickISR( )和OSIntCtxSw( )。下面将讲述这5个函数的主要功能和移植。在移植中,我们主要通过CONTEXT_SAVE宏和CONTEXT_RESTORE宏,完成对寄存器的值的入栈和出栈操作。

1.1 OSTaskStkInit( )

这个函数主要完成了初始化任务堆栈的工作。因此,初始化完成后,任务的堆栈就是像才发生了中断一样,所以寄存器的值,函数的入口地址(这里与中断不一样,中断是将中断发生的下个地址保存在堆栈中)都保存在堆栈之中。

在μC/OS-Ⅱ操作系统中,任务的一般形式都是"Task(void * pdata)"这种形式。在这种形式中,都将一个指针作为一个函数的参数传入这个任务中。因此在OSTaskStkInit( )这个函数中一般都会把这个指针的值压入堆栈中。但是在C55X DSP中,当函数调用的时候,对于指针参数一般是将指针的值保存在AR0中传给子函数。因此在这里我们省去了对pdata

收稿日期:2006-02-23

作者简介:鲍劼(1980-),女,浙江奉化人,中国矿业大学图书馆助理馆员,研究方向:计算机应用。

这个指针参数的入栈。这里我们还应该注意到,由于是模仿一次中断,因此对于堆栈的格式我们应该与实际上 C55X DSP 发生中断的情况一致。

其主要代码如下:

```
_OSTaskStkInit:
MOV SP, mmap(AC1L) ;将堆栈 SP 的地址保存到 AC1L 中
MOV SSP, mmap(AC1H) ;将系统堆栈 SSP 的地址保存到 AC1H 中
ADD #-20, AR1, AR2 ;通过 AR2 分配这个任务的数据堆栈的地址
MOV AR2, SP ;将分配的数据堆栈的地址移入 SP 中
MOV AR1, SSP ; 建立系统堆栈
;模仿中断的自动上下文切换
PSH mmap(ST2_55);
PSH mmap(ST1_55);
PSH mmap(AC0L);
MOV mmap(ST0_55), T1;
MOV #0, *AR1-;
MOV T1, *AR1-;
MOV #0, *AR1-;
MOV #0, *AR1;
MOV AR1, SSP;
CONTEXT_SAVE;
MOV SP, AR0 ;
;返回进入程序是保存的数据堆栈 SP 和系统堆栈 SSP
MOV mmap(AC1L), SP;
MOV mmap(AC1H), SSP
RET;
```

1.2 OSStartHighRdy( )

这个函数只是一次,就是在 OSStart( )(操作系统开始进程调度)的时候调用这个函数。这个函数用来使就绪队列中优先级最高的任务开始运行。

通过下面这个函数 OSCtxSw( ),这时系统就可以跳转到当前系统优先级最高的这个任务中来。

其代码如下:

```
_OSStartHighRdy:
CALL #_OSTaskSwHook ; call _OSTaskSwHook
MOV #1, *(_OSRunning) ; 将 OSRunning 设置为 1
MOV *(#_OSTCBHighRdy), AC0;将 OSTCBHighRdy 指针存储到 AC0 中
MOV AC0, AR0 ; AR0 获得该任务 TCB 的地址
MOV *AR0(0), AC0 ; 将任务 SP 的地址存入 AC0 中
MOV mmap(AC0L), SP ; 装载数据堆栈
CONTEXT_RESTORE;
RETI; run task
```

1.3 OSCtxSw( )

由于该系统是一个实时抢占式系统,所以当一个优先级更高的任务处于就绪状态的时候,系统就需要进行任务切换。产生 C55X 系统的 27 号软件中断,然后在中断处理程序中调用该函数,完成任务的切换。

其代码如下:

```
CONTEXT_SAVE ; 存储寄存器堆栈
MOV SP, mmap(AC1L) ;
MOV *(_OSTCBCur), AC0;
MOV AC0, AR0 ;将现在堆栈的地址保存到 TCB 中
MOV AC1, *AR0(0);
CALL #_OSTaskSwHook;
; OSTCBCur = OSTCBHighRdy
; OSPrioCur = OSPrioHighRdy
MOV *(_OSTCBHighRdy), AC0;
MOV *(_OSPrioHighRdy), AC1;
MOV AC0, *(_OSTCBCur);
MOV AC1, *(_OSPrioCur);
; restoring the new task
MOV AC0, AR0;
MOV *AR0(0), AC0;
MOV mmap(AC0L), SP; 从新堆栈中返回数据堆栈的地址
NOP; 通过空转命令使上面的命令能够完全执行
CONTEXT_RESTORE;
RETI;
```

1.4 OSTickISR( )

该函数就是系统通过 C55X 芯片的计数器产生时钟节拍中断的一个中断服务程序。系统就是周期地产生时钟节拍中断来产生系统的时钟信号。

我们在使用对时间中断编码有一种比较简单的方法,而且不需要修改源代码。就是在用 C 语言对这个中断进行编写后,然后对这个 C 语言编写的文件进行汇编,就得到了这个时钟中断的汇编程序。

1.5 OSIntCtxSw( )

该函数的执行一般都是在产生时钟中断的时候执行。其主要的功能就是当系统中由于时延的减小,而有优先级比当前任务的优先级高的任务进入就绪队列。这时就需要在中断服务程序中找到这个高优先级任务的 TCB,获得该任务堆栈地址,并且恢复到 C55X DSP 的寄存器中。由于 OSIntCtxSw( )函数和 OSCtxSw( )函数的主要功能基本一样。只是由于在时间中断服务程序中,已经保存了系统寄存器的值,这里就不需要该函数保存当前寄存器的值。所以这里就不在进行复述。

2 程序测试

仿真程序是在 CCS 下面完成仿真的。该测试程序创建了 2 个优先级不同的任务 A、B,(下转第 45 页)

```
"";
    currentPos = obj;
    send_request("ajax.jsp? playPos=" + obj);}
```

最后就是服务器端的 jsp 代码了：

```
<%@ page contentType="text/html; charset=gb2312" errorPage="" %>
<%String playPos = request.getParameter("playPos");
if("pos_1".equals(playPos))
out.print("  南昌市 <br>   九江市 <br>   赣州市");
else if("pos_2".equals(playPos))
out.println("  日照市 <br>   济南市 <br>   烟台市");%>
```

5 结束语

利用 Ajax，开发人员能够创建一个与桌面应用程序相媲美的交互式用户界面。Ajax 的最大挑战并不是技术，组成 Ajax 的各项技术已经相当成熟、稳定及易学，主要挑战是如何使用 Ajax 设计应用程序，我们必须抛弃以往对于 Web 局限性的看法，开阔视野、开放思想、不断创新，实现更加丰富的应用体验。

参考文献：

[1] [美]Jesse James Garrett. Ajax：A New Approach to Web Applications[DB/OL]. http://www.javalobby.org/articles/ajax,2005-02-18.

[2] Ryan Asleson, Nathaniel T Schutta. Ajax 基础教程[M]. 北京：人民邮电出版社，2006.

[3] 陈会安. Javascript 网页制作彻底研究[M]. 北京：人民邮电出版社，2002.

(上接第 32 页)其对应的优先级分别是 5、10。其中 B 任务运行 2 秒钟后发信号调用 A 任务的运行。A 任务运行 3 秒后结束，B 任务运行 5 秒结束。其中每个任务每一秒发出一个正在运行的标志。整个系统的仿真情况如下：

```
Task B is running ...
Task B is running ...
Task A is running ...
Task A is running ...
Task A is running ...
Task B is running ...
Task B is running ...
Task B is running ...
```

3 结束语

本文介绍了 μC/OS-Ⅱ在 TMS320VC5509 DSP 上面的移植，通过仿真测试程序，证明了移植的成功。对于以后在 TMS320VC5509 DSP 上程序的开发只需要在该操作系统上面进行，这样大大提高了开发的效率。

参考文献：

[1] TI. TMS320C55x Optimizing C/C++ Compiler User's Guide [Z]. 2002.

[2] Jean J Labrosse. 嵌入式实时操作系统 μC/OS-Ⅱ(第 2 版)[M]. 邵贝贝，等译. 北京：北京航空航天大学出版社，2003. 73～74.

[3] 汪春梅，孙洪波，任治刚. TMS320C5000 系列 DSP 系统设计与开发实例[M]. 北京：电子工业出版社，2004. 8～63，192～216.

(上接第 42 页)细节。交易 Agent 在知识库中记录这些细节，显示在信息流 ACI-5(把交易请求发给供应商)。交易双方通过信息中间商的交易 Agent 确认交易，这保证了接下来的实物或服务的交易。这些通信为图 1 的 AC-Ⅴ。

(6)后期交易信息收集。交易双方必须告诉交易 Agent 他们对交易的满意度。这由图 1 中 AC-Ⅵ表示。交易 Agent 在信息中间商的知识库中记录这个交易满意度(通过信息流 ACI-Ⅵ)。这些信息用来建立购买商和供应商的信誉档案以简化基于信誉的机制，这对提高市场的成功很重要。

3 结束语

在基于 Internet 的供应链管理中，信息的无二义性和如何进行协作是很复杂的问题。本文将分布式人工智能的多 Agent 技术应用于供应链管理，介绍了一种基于多 Agent 的电子供应链的体系结构。

参考文献：

[1] 史忠植. 智能主体及其应用[M]. 北京：科学出版社，2000.

[2] 马世华，林勇，陈志祥. 供应链管理[M]. 北京：机械工业出版社，2001.

[3] Berenji Dr H R, Vengerov D. Learning, Cooperation, and Coordination in Multi-agent Systems[R]. Intelligent Inference Systems Corp. Technical Report IIS-00-10, 2000.

[4] Gruber T. A translation approach to portable ontology specification[J]. Knowledge Acquisition, 1993,5(2).