基于COM的嵌入式系统通用硬件抽象层框架设计

王力生,孔庆雯
(同济大学 计算机科学与工程系,上海 200433)

摘 要:随着嵌入式系统的飞速发展,各种嵌入式处理器以及片上系统(System on Chip, SoC)应用于各种设备,从传感器、手机、PDA等到笔记本电脑。嵌入式系统的广泛应用促进了嵌入式软件,特别是嵌入式操作系统的发展,但嵌入式系统硬件体系结构的多样性又给嵌入式软件、嵌入式操作系统的开发、维护带来了极大的不便。文中针对硬件平台依赖性制约嵌入式操作系统发展问题,提出了一种应用COM技术开发的硬件抽象层设计,实现嵌入式操作系统跨硬件平台移植。

关键词:嵌入式操作系统;硬件抽象层;COM技术
中图分类号:TP303 **文献标识码**:A **文章编号**:1673-629X(2008)08-0242-04

Design of Embedded General Hardware Abstraction Layer Framework Based on Microsoft COM

WANG Li-sheng, KONG Qing-wen
(Computer Science and Engineering Department, Tongji University, Shanghai 200433, China)

Abstract: With the development of embedded system in recent years, embedded processor and SoCs are used in many devices, ranging from sensors, phones and PDAs, to general-purpose laptops. The spreading of embedded system has promoted the development of embedded software, especially of embedded operating system. In despite of this, the multiformity of the embedded system hardware architecture brings discommodity to the developing and maintenance of embedded software, especially for embedded operating system. In this article, present a new embedded hardware abstraction layer, using Microsoft COM, aiming at hardware platform dependency. With this HAL(hardware abstraction layer), will achieve the cross-platform transplantation of embedded operating system.

Key words: embedded operating system; hardware abstraction layer; Microsoft COM

0 引 言

随着应用的扩大和软硬件技术的发展,嵌入式系统逐渐从简单的单片机系统发展成高性能嵌入式微处理器和嵌入式操作系统应用阶段。操作系统已经成为嵌入式系统的核心。获得嵌入式操作系统最简便的方法是根据自己的需求对通用嵌入式操作系统进行功能上的裁减,并根据硬件平台进行修改。但用此方法获得的目标系统未必能够适合嵌入式应用。另一种方法是按照嵌入式应用的要求,针对底层硬件平台设计专用的嵌入式操作系统。该方式开发的目标系统能较好地适应嵌入式应用的需要。但缺点是因用时长、成本高、强硬件相关性而难以移植或复用[1]。

嵌入式系统的结构逐渐由三层演化为四层。新增中间层叫做硬件抽象层(Hardware Abstraction Layer, HAL)[2],位于操作系统和硬件平台之间,代表了逻辑上经过抽象的硬件环境,包含了系统中与硬件相关的大部分功能。通过特定的上层接口与操作系统进行交互,操作系统屏蔽底层的硬件特性,并根据操作系统的要求完成对硬件的直接操作。硬件抽象层的引入大大推动了嵌入式操作系统的通用程度,从而为嵌入式操作系统的广泛应用提供了可能。同时考虑到软件技术盛行的复用概念,采用组件技术来开发硬件抽象层是明智之举[3]。

收稿日期:2007-11-19
作者简介:王力生(1956-),男,上海人,副教授,研究方向为嵌入式系统及其应用。

1 嵌入式操作系统内核分析

对嵌入式操作系统的内核进行研究,分析出各个功能模块,是进行组件化硬件抽象层设计的关键前提。内核是嵌入式操作系统的必备基础。它提供任务管理、内存管理、通信/同步与互斥机制、中断管理、时间管理及任务扩展等功能。内核还提供特定的应用编程接口,但目前没有统一的标准[4]。

1.1 任务管理

任务管理是内核的核心部分,具有任务创建、任务

调度、任务删除、任务挂起、任务唤醒以及任务优先级设置等功能。嵌入式操作系统多采用基于静态优先级的抢占式调度。

1.2 内存管理

嵌入式操作系统的内存管理比较简单,通常不采用虚拟存储管理,而采用静态内存分配和动态内存分配相结合的管理方式。一些高端的嵌入式操作系统利用 MMU(Memory Management Unit)机制提供内存保护功能。

1.3 通信,同步和互斥

这些机制提供任务间以及任务与中断处理程序间的通信、同步、互斥功能,一般包括信号量、消息、事件、管道、异步信号和共享内存等功能。与通用操作系统不同的是,嵌入式操作系统需要解决在这些机制的使用中出现的优先级反转问题[2]。

1.4 中断,异常管理

中断管理具有以下功能:(1)安装中断服务程序;(2)中断发生时,对中断现场进行保护,并跳转到相应的服务程序上执行;(3)中断堆栈切换;(4)中断退出前进行任务调度;(5)中断退出前,对中断现场进行恢复。

中断管理负责控制中断控制器,负责中断现场的保护和恢复。为防止堆栈的溢出,专门设置中断栈。一旦进入中断就切换到中断栈,退出中断时再进行堆栈的切换。

1.5 时间管理

时间管理提供高精度、应用可设置的系统时钟。

1.6 任务扩展功能

嵌入式系统的应用广泛,但功能不可能面面俱到。任务扩展功能在内核中设置 Hook 调用点,在这些调用点上应用自己编写的扩展处理程序,以扩展内核功能[5]。

1.7 电源管理

内核的电源管理模块具有两大功能:基本的电源管理机制,应用系统操作电源管理的应用编程接口(API)。为满足嵌入式系统低功耗的要求,CPU 和外围设备大都提供了可编程控制的多功耗工作模式[5]。

2 通用硬件抽象层框架的设计

通用硬件抽象层的设计,是为不同嵌入式硬件平台上的嵌入式操作系统内核设计开发提供统一的硬件相关的功能。这套通用硬件抽象层框架包含 7 个模块,见图 1。

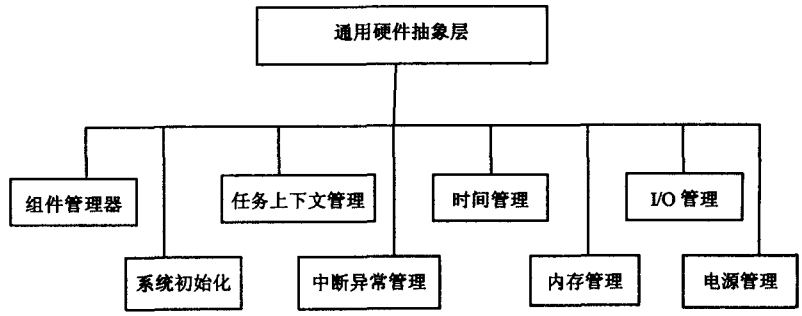

图 1　通用硬件抽象层的框架

2.1 系统初始化模块

该模块有 4 个接口:地址空间重映射,堆栈寄存器初始化,必要硬件初始化和建立 C 语言运行环境。

(1)地址空间重映射接口:针对有 MMU 的处理器,地址空间的重映射需要建立页表并设置 MMU,对于没有 MMU 的嵌入式处理器,则是通过对系统控制寄存器的操作来进行的。

```
[object,uuid(INTERFACEID)]//INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IAddressReMap{
HRESULT PageTable([out] AddressType * pat);//建立页表,AddressTpye 为系统中地址类型,返回一个指向页表的指针
HRESULT SetMMU(void);//设置 MMU
HRESULT SetSysReg(void);//设置系统控制寄存器
}
```

(2)堆栈寄存器初始化接口:在地址空间重映射后,对 CPU 各种堆栈指针寄存器进行初始化,分配 RAM 空间,为中断和异常分配有独立的栈空间。

```
[object,uuid(INTERFACEID)] //INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IStackReg{
HRESULT InterruptStack([out] StackType * pis);//初始化中断/异常堆栈空间,返回指向栈顶的指针
HRESULT SetContrReg ([out] Reg globinter, [out] Reg mode);//设置 CPU 控制寄存器,两个参数分别为寄存器型的全局中断始能位和寄存器型的模式
}
```

(3)必要硬件初始化接口定义:

```
[object,uuid(INTERFACEID)] //INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IHardwareInitial{
HRESULT SetFrequency([out] Time freq);//设置时钟频率,参数为时间类型的频率
HRESULT SysContr(void);//初始化系统控制
HRESULT BusContr(void);//初始化总线控制
HRESULT InterContr(void);//初始化中断控制器
HRESULT SetTime(void);//初始化定时器/计数器
}
```

(4)建立 C 语言运行环境接口:C 语言的运行环境的建立是通过创建初始值为 0 的未初始化数据段.bss 段来实现的。

```
[object,uuid(INTERFACEID)] //INTERFACEID 为此接口的 uuid,此时还不能确定
Interface ISetC{
HRESULT Setbss(void);//为运行 C 语言初始化.bss 段
}
```

2.2 任务上下文管理模块

该模块有 4 个接口:任务上下文创建,直接任务上下文切换,间接任务上下文切换,任务上下文加载。

(1)任务上下文创建接口:嵌入式 OS 内核在创建一个任务时需在任务堆栈中创建任务上下文。该任务被调度上台执行时从其堆栈中将任务的上下文恢复到 CPU 的寄存器,以实现任务的上台。

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IContexCreate{
HRESULT ContexCreate([in] AddressType * ptr_t);//返回上下文建立后任务堆栈指针内容
}
```

(2)直接任务上下文切换接口定义:

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IContexSwitch{
HRESULT ContextSwitch(void);
}
```

(3)间接任务上下文切换接口定义:

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IIntContexSwitch{
HRESULT IntContextSwitch(void);
}
```

(4)任务上下文加载接口:当系统任务被调度执行时,需要将该任务的上下文加载到 CPU 的寄存器中开始运行。

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IContexLoad{
HRESULT ContextLoad(void);
}
```

2.3 中断异常管理模块

该模块有 3 个接口:中断向量表管理,中断管理和异常管理。

(1)中断向量表管理接口:负责对中断异常向量表入口项的管理,为异常管理和中断管理,甚至是操作系统内核直接提供统一的向量表入口项安装中断异常处理程序的服务。

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IInterruptTable{
HRESULT GetVectorHandler([in] AddressType * handler-t);//获得对应入口项处理程序地址
HRESULT SetVectorHandler(void);//设置向量表入口项
}
```

(2)中断管理接口:主要功能有中断管理系统初始化,中断请求优先级设置,中断请求使能与禁止,请求抽象 IRQ 设备,释放抽象 IRQ 设备,内核中断内调度程序注册,中断嵌套入口程序注册,中断嵌套出口程序注册,关全局中断并返回原来的全局中断状态,恢复原来的全局中断状态。

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IInterruptManage{
HRESULT InitialInterrupt(void);//初始化中断管理系统
HRESULT SetIrqPrio([in] long n);//设置中断请求优先级,返回 3 种情况(成功,中断请求号无效,中断优先级无效)
HRESULT EnableIrq([in] long err_t);//中断使能,返回有 2 种情况(成功,中断请求号无效)
HRESULT DisableIrq([in] long err_t);//中断禁止,返回有 2 种情况(成功,中断请求号无效)
HRESULT RequestInterrupt([in] long err_t);//申请抽象 IRQ 设备,返回 2 种情况(成功,中断请求号无效)
HRESULT FreeInterrupt([in] long err_t);//释放抽象 IRQ 设备,返回 2 种情况(成功,中断申请号无效)
HRESULT SetIntExit(void);//内核中断内调度程序注册,用于注册中断返回前的内核调度程序
HRESULT SetBeforeISR(void);//该方法用于注册在执行外设中断服务程序之前进行的执行中断嵌套必需的保护工作的程序。
HRESULT SetAfterISR(void);//该方法用于注册在执行外设中断服务程序之后进行的执行中断嵌套必需的恢复工作的程序
HRESULT DisableInterrupts([in] State reg_t);//返回原来的全局中断状态
HRESULT RestoreInterrupts(void);//恢复原来的全局中断状态
}
```

(3)异常管理接口:由于受不同体系结构对异常种类定义各不相同的限制,异常管理部分采用在各种体系结构层中对各种异常处理进行包装,并提供对应异常的回调处理。

```
[object,uuid(INTERFACEID)]// INTERFACEID 为此接口的 uuid,此时还不能确定
Interface IExceptionmanage{
HRESULT ExceptionSwiHandler(void);//陷入处理回调
HRESULT ExceptionPrefetchInsHandler(void);//指令预取中止异常处理回调
HRESULT ExceptionPrefetchDatHandler(void);//数据访问中止异常处理
```

```
HRESULT ExceptionUndefinedHandler(void);//未定义指令异常处理回调
}
```

2.4 时间管理模块

定时管理接口类有8个接口,分别是初始化定时器接口;创建定时器接口(返回3种情况:操作成功返回定时器编号,定时器号无效,定时时间无效);释放抽象定时器接口(返回2种情况:成功,定时器号无效);启动抽象定时器接口(返回2种情况:成功,定时器号无效);获取抽象定时器接口(返回2种情况:抽象定时器状态,定时器号无效);设置抽象定时器接口(返回3种情况:成功,定时器号无效,定时器模式错误);获取抽象定时器周期接口(返回周期数);设置抽象定时器周期接口(返回3种情况:成功,定时器号无效,定时时间无效)。

2.5 I/O管理模块

I/O管理接口类有3个接口:驱动程序地址表管理接口(有3个方法:初始化表,登录所有驱动程序,返回表头指针;根据主设备号定位驱动程序地址,返回驱动程序地址;更新表,当安装新的驱动时,将其登录在表中);设备名表管理接口(有3个方法:初始化设备名表,返回表头指针;搜索设备名表 返回对应搜索设备的表项;更新表);文件描述符表管理接口(有3种方法:初始化表,返回表头指针;搜索表;更新表)[6]。

2.6 组件管理器模块

该硬件抽象层框架应用组件(COM)技术,把操作系统内核的各个功能抽象成功能模块,封装成组件。组件管理器用于管理组件,由组件登录表和组件管理表两部分组成。

凡在组件登录表中登录的组件,即在系统中注册,系统可以调用这些组件;并控制其生命周期,实现目标系统的动态性,当新的组件加入系统,只需在表中为其分配一个表项,并填写相关信息;当要剥离一个组件时,只需将对应表项删除即可。

组件登录表登录了系统中的所有组件,但并非每个组件都是活动的。当不需要某个模块时,就将其设为"休眠",节省电能。组件管理表中的每一项对应系统中一个处于"活动"状态的组件,并附有活动组件和对活动组件进行调用所需的信息。在系统中激活一个组件,就把组件登录表中的对应项写到组件管理表中,并补充完整信息。当要把某个组件设为"休眠"时,只要取消组件管理表中对应项,但保留组件登录表中的项。

3 结束语

针对嵌入式操作系统开发现在面临的诸多挑战,介绍了一种基于组件技术的通用硬件抽象层的框架,为嵌入式操作系统内核的设计开发屏蔽硬件平台的特性,提供统一的硬件相关的服务接口,使其设计开发可不依赖特定的硬件平台,并且具有更强的可移植性。

在设计通用硬件抽象层时使用组件的技术,将操作系统内核的功能抽象为各个功能模块,并封装为组件,这无疑又为快速、动态地搭建嵌入式操作系统提供了一条可行之路。

参考文献:

[1] Labrosse J. 嵌入式系统构件[M].第2版.袁勤勇,黄绍金,唐 青译.北京:机械工业出版社,2001.

[2] Massa A. Embedded Software Development with eCos[M]. [s.l.]:Prentice Hall, 2002.

[3] Cai Xia, Lyu M R. Component-Based Embedded Software Engineering: Development Framework. Quality Assurance and A Generic Assessment Environment[J]. International Journal of Software Engineering and Knowledge Engineering, 2003,12(2):107-133.

[4] Ford B. The Flux OSKit. A Substrate for Kernel and Language Research[C]//Proc. 16th ACM Symp. Operating Systems Principles. New York:ACM Press,1997:38-51.

[5] Friedrich, Stankovic. A Survey of Configurable, Component-Based Opreating-systems for Embedded Applications[J]. IEEE Micro, 2001,21(3):54-68.

[6] Tanenbaum A S. 现代操作系统[M]. 陈向群译. 北京:机械工业出版社,2000.

(上接第241页)

算机科学,2003,30(5):90-93.

[3] 杨芙清,梅 宏,李克勤,等.支持构件复用的青鸟III型系统概述[J].计算机科学,1999,26(5):50-55.

[4] Krut R, Zalman N. Domain Analysis Workshop Report for the Automated Prompt Response System Domain[R]. [s.l.]: Carnegie Mellon University,1996.

[5] 张 荣.面向领域的CASE与MIS开发[J].计算机应用研究,1998,4:71-74.

[6] STARS Technical Committee. Asset library open architecture framework version 1.2[M]//Software Technology for Adaptable, Reliable Systems (STARS). [s.l.]:[s.n.],1992.

[7] 李景峰,李 琰,陈 平.一种特定领域的软件开发模型[J].西安电子科技大学学报:自然科学版,2000,27(5):602-606.