第 35 卷 第 9 期 计 算 机 工 程 2009 年 5 月
Vol.35 No.9 Computer Engineering May 2009
·博士论文· 文章编号：1000—3428(2009)09—0014—03 文献标识码：A 中图分类号：TP311;TP39

面向对象的嵌入式系统电源管理模型

李豫东[1,2]，任建岳[1]，金龙旭[1]
(1. 中国科学院长春光学精密机械与物理研究所，长春 130033；2. 中国科学院研究生院，北京 100039)

摘 要：针对嵌入式系统的低功耗设计，提出一种面向对象的软件电源管理模型。在操作系统的基础上将相关 API 封装与扩展，抽象出操作系统电源管理类(OSPM)、CPU 电源管理类(CPUPM)、设备驱动电源管理类(DDPM)、应用程序电源管理类(APM)。在操作系统层、驱动程序层、应用程序层之间形成电源管理接口，简化嵌入式系统电源管理的软件设计与维护。

关键词：嵌入式系统；电源管理；面向对象

Object-oriented Power Management Model of Embedded System

LI Yu-dong[1,2], REN Jian-yue[1], JIN Long-xu[1]
(1. Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

【Abstract】Aiming at low power consumption design of embedded system, a software power management model is put forward, which based on object-oriented principle. On the foundation of operating system, the related APIs are packed and expended. In the mean time, four classes are abstracted which are Operating System Power Management class(OSPM), CPU Power Management(CPUPM) class, Device Driver Power Management(DDPM) class and Application Power Management(APM) class. Within operating system level, device driver level, application level, and the power management interface are formed which make the software of power management easy to design and defend for embedded system.

【Key words】embedded system; power management; object-oriented

1 概述

越来越多的嵌入式设备采用电池供电，如手机、PDA、数码相机、掌上仪器等。随着这些设备性能的不断提高以及功能的不断扩展，对电池的要求也越来越高。如果一台数码相机的电池只能工作很短的时间，即使功能多也无法充分享用[1]。因此，很有必要利用高效的电源管理系统降低设备的功耗。

目前普遍采用动态电压扫描(DVS)与动态频率扫描(DFS)方法实现电源管理，着眼于减小 CPU 的功耗。由于 CPU 一般都提供内核电压与频率的动态调节功能，针对不同应用，在维持程序正常执行的前提下，减小 CPU 的内核电压与工作频率可以降低功耗[1-3]。这些方法虽然可以部分的降低 CPU 的功耗，但需要复杂的预测算法来确定电压与频率，往往忽略了对外部设备采取低功耗措施，因此，对系统整体功耗的降低是有限的。多数基于硬件设备的低功耗设计方法都具有一定的局限性。而操作系统、驱动程序与应用程序包含了硬件运行时的所有特征，如果能在软件平台的基础上应用高效率的软件电源管理系统，对软硬件资源进行合理的调度，就能最大限度地降低系统功耗[4]。

2 基于 API 的软件电源管理系统

一些功能强大的嵌入式操作系统内部通常嵌入了电源管理模块，并且以 API 调用的方式实现电源管理系统。电源管理需要内核、设备驱动、应用程序 3 个部分协同工作来实现。下文以 Linux 的动态电源管理系统(DPM)为例来说明。

内核中的 DPM 子系统负责维持系统的电源状态，并将 DPM 系统的各个得到管理的硬件设备联系在一起。DPM 子系统通过多个 API 与设备驱动程序通信，这些 API 把驱动程序从完全运行状态转为各种电源管理的状态[4]。

设备驱动程序要比默认驱动程序具有更多“状态”：由外部事件通过各种状态来驱动设备，或通过来自内核 DPM 的回调来驱动设备，从而反映并遵循电源管理策略。驱动程序 API 还须驱动程序登记各个设备的基本运行特征，从而实现更精确的电源管理决策。

应用程序可感知电源管理，能充分利用来内核与驱动程序中的 API，从而建立各自的电源状态约束，并强制电源管理状态发生变化，通过多个 API 向 DPM 子系统提供指导，在定义好的状态切换点之间转移整个系统。

针对以上 3 个部分，实现 Linux 的 DPM 首先要准确定义设备的电源管理状态，例如将电源管理状态划分为：系统复位状态，全速运行状态，空闲状态，挂起状态等。对于不同的处理器以及不同的外部设备，状态的定义通常是不一样的。其次要根据不同的电源管理状态，为外部设备编写电源管理 API。再次开发应用程序时必须添加可感知电源管理的部分，定义与内核 DPM 之间的接口 API，可感知系统当前的电源管理状态，根据具体应用确定下一状态，并及时通知 DPM 改变 CPU 与相关设备的电源状态[5-7]。

由于嵌入式设备的功能与外设越来越丰富，这种 API 调用方式的电源管理系统很复杂，因此驱动程序与应用程序的

基金项目：国家自然科学基金资助项目(60403025)
作者简介：李豫东(1982－)，男，博士研究生，主研方向：嵌入式系统，图像处理；任建岳、金龙旭，研究员、博士生导师
收稿日期：2008-08-19 **E-mail：**lydong555@126.com

开发与维护较为困难。如果用面向对象的原理，将相关的 API 进行封装与扩展，就可以形成面向对象的软件电源管理框架，还可以根据不同的配置与应用进行派生，从而简化电源管理系统的开发。

3 面向对象的电源管理系统

本文提出的面向对象的电源管理系统是一种参考模型，在嵌入式操作系统 API 的基础上，抽象出操作系统电源管理类(Operating System Power Management class, OSPM)，CPU 电源管理类(CPU Power Management, CPUPM)，设备驱动电源管理类(Device Driver Power Management, DDPM)，应用程序电源管理类(Application Power Management, APM)，它们之间的作用机制如图 1 所示。

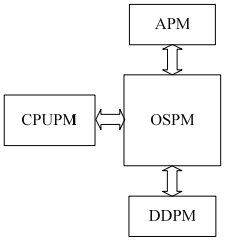

图 1 OSPM, CPUPM, APM, DDPM 之间的作用机制

3.1 操作系统电源管理类

OSPM 是电源管理系统的核心，是集成于操作系统内核的电源管理策略，在收集系统各部分电源管理信息的基础上，对系统中所有的软硬件资源进行合理调度，使不同应用程序与外部设备以尽可能低的功耗运行。

由图 1 可见，OSPM 就像 DDPM, APM 与 CPUPM 之间的中介，OSPM 接受 APM 的电源管理请求(*Ack_AppRequest*())，然后通过 CPUPM 得知 CPU 当前的电源状态(*Get_CpuState*())，根据 APM 的具体需求，OSPM 通知 CPUPM 调整或改变 CPU 的电源状态(*Change_CpuState*())；OSPM 也收集各个外设器件的电源状态(*Get_SystemState*())，根据当前应用，将各个外设器件配置为正常运行所需最低功耗的电源状态(*Set_SystemState*())；OSPM 还负责监测电池状态(*Get_BatteryState*())，当电池供电不足时，OSPM 将强制某些外设进入低功耗状态或禁止外设(*Set_LowPowerMode*())，产生如显示屏背光变暗，屏保等待时间缩短等现象，并通知某些应用程序改变运行模式(*LowPowerNotify*())，禁止某些应用(如声音与视频)。

3.2 CPU 电源管理类

图 2 为 CPUPM 中的状态定义与状态转换机制。

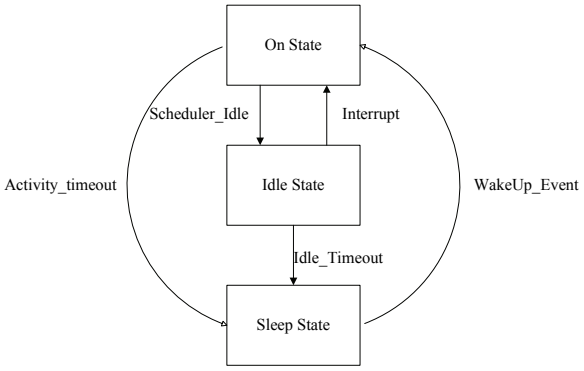

图 2 CPUPM 的状态

CPU 的状态概括为以下 3 种：(1)On State，系统处于通常的操作状态，此时针对 CPU 的动态电压扫描与动态频率扫描起主要作用(*Set_CoreVoltage*(), *Set_CoreFrequency*())；(2)Idle State，此时为可快速唤醒状态，系统的运行环境保持在 CPU 中(*Maintain_Context*())，CPU 保持低功耗的上电状态，一旦发生外部中断，CPU 立即返回 On State(*ResumeOn*())，当空闲定时器超时，CPU 将进入 Sleep State(*Set_Sleep*())；(3)Sleep State，系统的运行环境被保持在内存中(*Store_Context*())，CPU 处于掉电状态，当有唤醒事件发生时，CPU 将返回 On State。

CPUPM 类与 OSPM 类的作用机制如图 3 所示，CPUPM 接受来自应用程序的电源请求后(*Acknowledge_App*())，获取 CPU 的当前电源状态(*Get_CpuState*())，通知 CPU 调整内核电压与频率，或利用中断与唤醒的方式通知 CPU 改变当前电源状态，CPUPM 则用动态调节的方式，或通过状态转换机制(*Set_CpuState*())完成电源状态的改变。

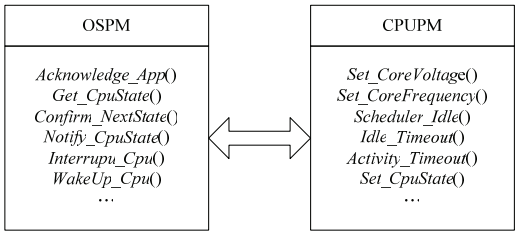

图 3 OSPM, CPUPM 之间的作用机制

3.3 设备驱动电源管理类

DDPM 定义了设备驱动程序的电源管理框架，并包含了外设器件可能具有的所有电源管理状态，根据不同外设的功耗特征，可派生出许多子类，如 KeyboardPM, DisplayPM, StoragePM, CameraPM 等。DDPM 还要求外部设备拥有即插即用功能，并根据具体应用及时进入空闲状态，在需要恢复时，硬件可被及时唤醒，在供电不足等特殊情况下，硬件设备可转变运行模式或被禁止，如图 4 所示。

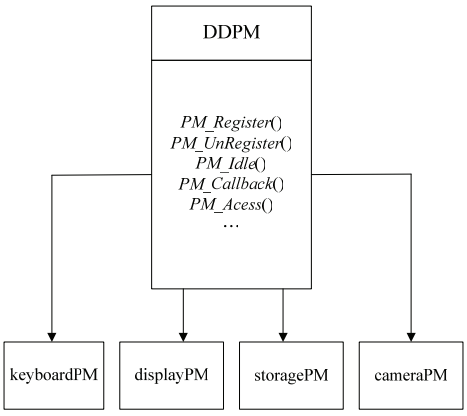

图 4 DDPM 及其派生类

图 5 为 DDPM 与 OSPM 的接口作用关系，外设插入与拔出时，DDPM 先向 OSPM 注册或注销(*PM_Register*(), *PM_Unregister*())，OSPM 监测系统电池的状态后，返回一个确认信息(*Ack_Request*())；进行电源管理动作时，DDPM 根据需要对硬件设备进行相应操作(*PM_Acess*(), *PM_Idle*()或 *PM_Callback*())，返回设备状态(*DevStateNotify*())，OSPM 读取原系统设备状态信息(*Get_SystemState*())，根据电源管理需要重新设置系统设备状态信息(*Set_SystemState*())；电池电量不足时，OSPM 设置低电量模式(*Set_LowPowerMode*())，强

制某些设备改变运行模式或禁止某些设备。

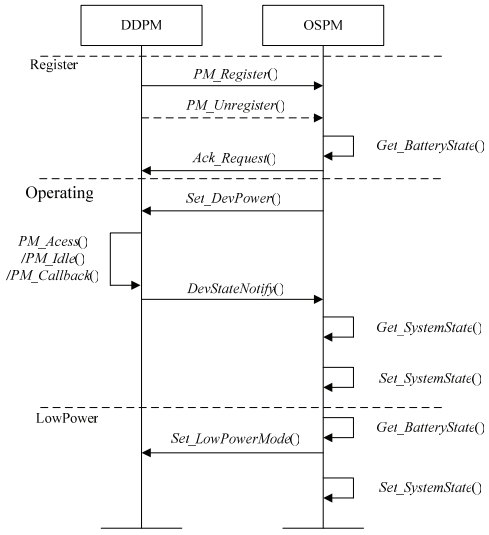

图 5　OSPM, DDPM 之间的作用关系

3.4　应用程序电源管理类

APM 定义了应用程序的电源管理框架，包含了使应用程序能感知电源管理的部分，包括应用程序电源管理请求 *App_PmRequest*() 和应用程序硬件设备请求 *App_DevRequest*()，可派生出许多应用程序电源管理子类，如 voicePM, videoPM, gamePM, emailPM 等。图 6 为 APM 与 OSPM, DDPM 的相互作用关系。

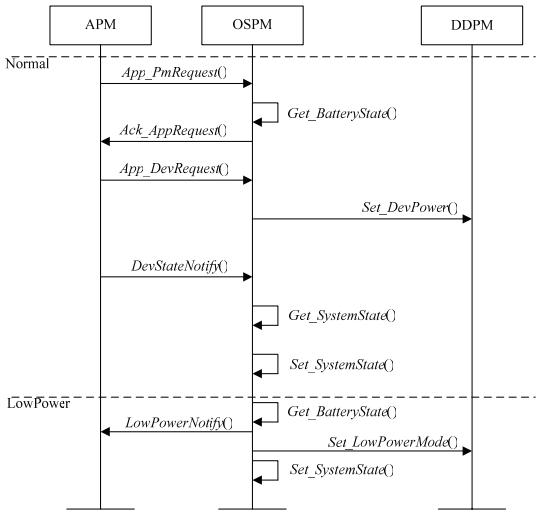

图 6　APM, OSPM 之间的作用关系

一个应用程序开始运行时，APM 先向 OSPM 发出电源请求，OSPM 监测电池状态后，返回确认信息(*Ack_AppRequest*())，APM 再发出相应的设备请求，OSPM 设置相应设备的电源状态(*Set_DevPower*())，并读取原系统设备状态信息(*Get_SystemState*())，然后重新设置系统设备状态信息(*Set_SystemState*())。在电池电量不足的情况下，OSPM 向应用程序发出低电量通知(*LowPowerNotify*())，强制应用程序改变运行模式或禁止某些应用。

4　结束语

本文将面向对象原理引入嵌入式电源管理系统，提出一个供参考的面向对象的电源管理系统模型。面向对象的软件结构具有通用性，合理的模型不仅可应用在各种嵌入式操作系统的平台上，而且能构建出通用且简洁的电源管理接口，简化驱动程序与应用程序电源管理部分的设计，提高电源管理效率。

参考文献

[1] Sandy I, Sandeep S, Rajesh G. Online Strategies for Dynamic Power Management in Systems with Multiple Power Saving States[J]. ACM Transactions on Embedded Computing Systems, 2003, 2(3): 325-346.

[2] Bishop B, Karthick R. Dynamic Power Management for Embedded Systems[C]//Proc. of the IEEE International SOC Conference. Portland, Oregon, USA: [s. n.], 2003: 17-20.

[3] Olsen C M, Narayanaswarni C. An Efficient Power Management Scheme for Mobile Devices[J]. Mobile Computing, 2006, 5(7): 816-828.

[4] Olsen C M, Narayanaswami C. A Work Dependent OS Timing Scheme for Power Management: Implementation in Linux and Modeling of Energy Savings[R]. New York, USA: IBM Inc, Tech. Rep.: 22784, 2003.

[5] Daniel P, Macro C. Understanding Linux Kernel[M]. 2nd ed. [S. l.]: O'Reilly, 2002.

[6] IBM MontaVista Software. Dynamic Power Management for Embedded Systems[EB/OL]. (2002-04-03). http://www.Research.ibm.com//arl/projects/dpm.html.

[7] Qiu Qinru, Wu Qing, Pedram M. Dynamic Power Management in Mobile Multimedia System with Guaranteed Quality-of-Service[C]//Proc. of IEEE Design Automation Conference. New York, USA: ACM Press, 2001: 834-849.

编辑　金胡考

（上接第 13 页）

参考文献

[1] Littman M L. Markov Games as a Framework for Multi Agent Reinforcement Learning[C]//Proceedings of the 11th International Conference on Machine Learning. Los Alamitos, CA, USA: IEEE Press, 1994.

[2] 高　阳, 陈世福, 陆　鑫. 强化学习研究综述[J]. 自动化学报, 2004, 30(1): 88-102.

[3] Tsitsiklis J N. Asynchronous Stochastic Approximation and Q-learning[J]. Machine Learning, 1994, 16(3): 185-202.

[4] Vlassis N. A Concise Introduction to Multiagent Systems and Distributed AI[Z]. (2003-09-09). db.sis.pitt.edu/infsci3350/resources/cimasdai.pdf.

[5] Tan M. Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents[C]//Proc. of the 10th. Int'l Conf. on Machine Learning. Amherst, MA, USA: [s. n.], 1993.

编辑　张正兴