DOI:10.3969/j.issn.1001-4551.2013.02.001

机器人控制软件框架中实时任务模块的设计与实现

李晓明,孙辰晨
(浙江理工大学 机械与自动控制学院,浙江 杭州 310018)

摘要:针对机器人控制中存在的实时性要求,在原组件式机器人控制软件框架的基础上设计了新的任务模块接口。提出了实时任务模块的概念,采用实时时钟模型执行多实时任务的方法,实现了多任务的并发实时运行;并基于Java平台和Windows操作系统进行了实现,其中框架整体采用Java平台实现,Java平台中的时钟采用JNI技术在Windows操作系统上实现;最后进行了系统实时性测试,测试结果验证了该系统的实时性要求。研究结果表明:经过改进后的机器人控制软件框架,适合机器人控制软件的开发,尤其是对多任务实时性有一定要求的机器人控制软件开发;同时该控制软件框架降低了架构机器人控制软件的难度,并为其软件开发提供了一个柔性开发平台。

关键词:机器人;软件框架;实时性;实时任务;Java
中图分类号:TP242.6 **文献标志码**:A **文章编号**:1001-4551(2013)02-0129-05

Robot control module software framework for real-time tasks

LI Xiao-ming, SUN Chen-chen
(Faculty of Mechanical Engineering & Automation, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: Aiming at the real-time requirement of robotic control, some improvements were made based on the previously developed component based robotic software framework. These improvements included: the newly designed interface for real-time task, the concept of real-time task module, the new model for real-time clock, and new scheme for execution of multiple real-time tasks. The new framework was implemented using the Java programming language, and the real-time clock was implemented based on JNI, so that the Windows OS based multimedia clock could be integrated into the Java program and providing the most accurate clock event in the PC platform. The test result demonstrates the effectiveness of the framework. The results indicate that the improved software framework is more suitable for mechatronic control software development, especially for the application of robot, which has certain requirements of real-time performance. This software framework decreases the difficulties in building robotic control softwares and provides a strong flexible development for its software development platform.

Key words: robot; software framework; real-time; real-time task; Java

0 引 言

现代的机电系统已经朝着智能化、网络化、分布式的方向发展,其对软件的开发也提出了更高的要求。以机器人技术[1-4]为例,单个机器人本身就是一个复杂的机电系统,需要通过自身的软件设计来实现各种不同功能,适应不同的环境。课题组研究了适用于机器人此类复杂机电系统的软件开发方法和开发模式,提出了一种轻量级的基于组件的软件框架[5-8],可简化移动机器人应用程序的结构,使设计工作变得省时。该架构基于组件的软件工程,保证最大限度的模块化和可重用性。

收稿日期:2012-10-16

作者简介:李晓明(1976-),男,山东文登人,工学博士,副教授,硕士生导师,主要从事运动控制技术、机电智能技术等方面的研究. E-mail: origin2007@gmail.com

该框架最大的不足是系统的实时性问题。以机器人控制为例,在现有软件平台上,研究者尽管可以设计出各个功能模块,通过模块之间的相互协作和数据交换实现诸如导航、避障等控制任务,但由于其采用多线程实现机制,具体功能模块在实际运行时无法确定自己的运行时刻,这对于一些有实时性要求的应用场合来说是不符合其要求的。实时性要求系统必须对外来事件在限定时间内做出反应。例如,足球机器人的策略系统软件,其功能模块运行必须在两次图像采样之间完成,且功能模块的运行周期要求必须稳定和精确,否则诸如预测模块、运动控制、规划等模块就会失效。大部分机器人控制系统对实时性要求非常严格,各控制任务必须在一定的时间内完成,否则各任务之间会产生资源和时间上的竞争,造成控制程序产生错误指令、死锁甚至崩溃,对机器人的性能造成严重影响。所以机器人实时控制部分必须精确安排各任务的执行顺序和执行时间,避免上述情况发生。

本研究将在上述软件架构的基础上加入实时任务模块的设计,完善系统,使其可以实现机器人控制的实时性。

1 模块化机器人软件框架

软件整个框架的结构包括:框架的模型、模块的接口定义、模块的加载和配置、模块之间的通讯[9],即Platform平台、Module模块和Wire通讯连线等。

Platform类负责模块的加载、运行、停止以及销毁,分析模块的配置并建立模块之间的通讯。

在软件框架内部,以模型Module的概念来表示一个功能模块。一个软件系统由许多个Module组成,Module之间保证最大程度的独立性,每个Module可以独立运行于Platform上(以线程的方式),并通过端口与外部进行通讯。研究者通过合理划分Module的功能,以及不同模块之间通过通讯管道进行组合,可以实现不同的系统应用,这也是组件编程思想的主要部分。这种模型驱动编程机制源于Auslander[10-11]提出的基于任务和状态的机电系统软件模型以及多智能体的基本模型。

平台中组件之间的通讯采用了基于管道技术的通讯模型。管道资源由平台分配,组件在平台中注册名字唯一的通讯管道,就可以向管道中写入数据或者接收该管道中的数据。开发者采用这种方式可以方便地实现多个组件的协作。

该软件系统框架的优点是,开发者可以专注于组件的开发,并可复用已经开发的组件,通过XML配置文件对组件进行组合,实现自己所需要的具体控制应用。但由于每个组件是以单独线程的方式在运行,对于机器人控制系统而言,在任务这一层次上的管理较为困难;此外,每个模块功能的执行顺序依赖于线程的同步,整体执行效率比较低,而整个系统的控制周期是难以确定的,因此难以应用于对实时性有一定要求的机器人应用中。

2 软件框架中实时任务模块的设计

2.1 实时性需求

在复杂机电系统或者机器人等应用中通常会涉及到实时控制的问题,从简单的温度控制的应用到复杂的机器人视觉伺服应用等。由于存在着实时控制,要求研究者在开发这些软件的时候必须考虑实时性问题。

对于任何一个计算机控制系统而言,其控制系统设计的一个重要前提都是采样周期 T 。采样周期决定了控制系统的许多性能,甚至稳定性。同理,在实际控制中精确稳定的采样(控制)周期对控制系统的实际控制效果也同样非常重要。首先,如果在实际软件开发中不考虑控制周期的稳定性问题,那么在实际控制中由于采样周期的变化,原先设计的控制算法可能不会达到预想的控制效果,甚至失效;其次,即使考虑了控制周期的稳定性,但如果实际执行时控制周期与设计时的采样周期不一致,同样会导致控制系统效果变差或者失控。

以常用的PID控制器为例:

$$u(k)=k_{\mathrm{p}}[e(k)+\frac{1}{T}\sum_{i=0}^{k}e(i)+T_{\mathrm{d}}\frac{e(k)-e(k-1)}{T}]+u_0 \quad (1)$$

当 T 的值发生变化时,控制器的输出 u 也会发生变化,并最终影响系统的稳定性。因此,对于面向实时控制的机电系统软件框架而言,实时性需求应当包括两点:①能够提供准确的采样周期;②能够提供稳定的采样周期。

2.2 模块化实时软件框架设计

加入实时任务模块的设计后,软件框架的总体结构图如图1所示。

该设计将Platform升级为具有实时功能的运行平台,将组件升级为包含实时任务接口的组件模型,同时保留原组件模型为非实时组件。Platform平台将主要包括实时时钟设计、实时任务调度设计、组件管理模型设计、通讯模型设计等关键内容。其中,组件管理模型设计、通讯模型设计和原模型基本相同,下面着重介绍实时时钟和实时任务调度的设计。

(1) 实时时钟模块。为了解决控制中的实时性问题,本研究在平台模型中设置了实时时钟模块。实时

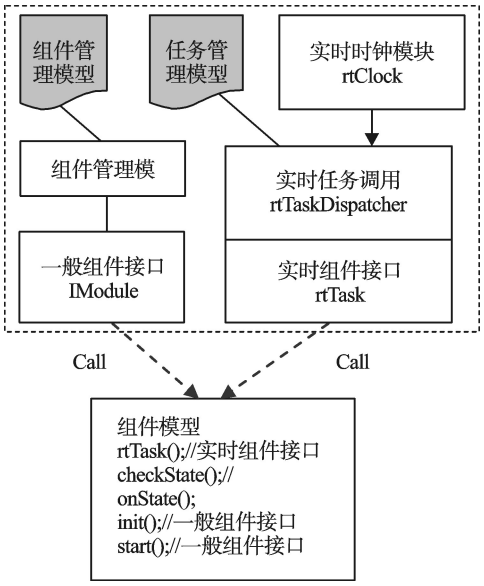

图1　具有实时功能的软件框架模型

时钟模块的实现在嵌入式系统中比较容易,但是在基于PC的系统上很难保证其绝对的实时性,尤其是运行于诸如Windows等非实时的操作系统上时。一般情况下,在PC上运行的控制系统属于监督控制范畴,即属于软实时系统,其实时性要求并非十分苛刻,因此可以采用宿主操作系统提供的时钟信号,例如在Windows平台下的多媒体时钟中断,可以提供精度为1 ms的时钟周期,可以满足大多数基于PC的控制需求。例如在基于PC 的运动控制系统中,当运动出现异常时,运动控制卡能自动进入保护状态,因此对于该类控制问题,精度为1 ms的采样周期完全符合要求。事实上,对于此类应用,精度更差的普通定时器往往也能满足要求[12]。但是对于工业上的一些实时性要求相对高的实时任务,Windows系统的多媒体扩展库MMSYSTEM.LIB提供的多媒体定时函数已经不能满足系统需要,例如点胶控制系统[13]。

(2) 实时任务调度模块。实时时钟模块产生精度为1 ms且指定时间间隔的时钟信号,实时任务调度模块负责当检测到时钟信号时通过实时组件的接口调用组件中的实时任务程序进行执行。在设计时,本研究要求这些实时任务程序中的代码应尽量做到高效,执行时间短;同时严格禁止在实时任务模块中执行非实时的工作。任务管理模型中采用了动态的数据结构来维护平台中注册的所有实时组件,并可以通过设计算法动态地更新不同组件实时任务的优先级和调用顺序,将重要的模块优先执行。实时任务调度模块严格监控每个任务的执行时间,会对执行时间过长的任务降低其优先级,如果时间片即将用完,将会停止执行后续任务,保证下一时钟信号来临的时候可以进行下一控制周期高优先级的任务。实时任务调度示意图如图2所示。

上述实时任务的实现方法与点胶控制系统中的方法大致相同,都是利用定时器产生一定周期的定时中断,在中断中执行实时任务。不同的是在点胶控制系统中,实时任务执行完毕后可自行结束中断;但本研究中实时时钟控制周期一旦确定,即使实时任务结束后仍有空闲的时间片,也不能结束当前中断。

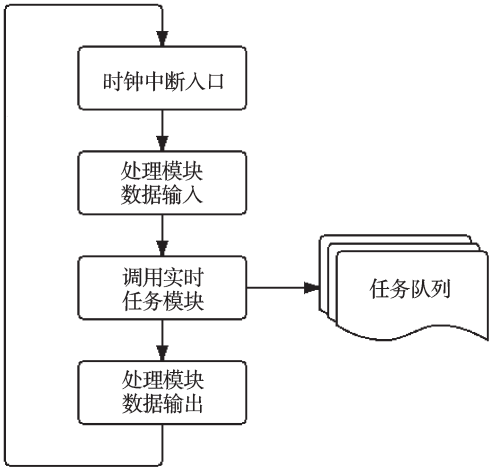

图2　实时任务调度

3　实时框架的实现

3.1　实时时钟模块的实现

本研究所设计的面向机电系统的软件框架是基于Java平台开发的,Java平台本身所提供的定时器显然无法满足前述实时性要求,因此必须采用操作系统所提供的精度更高的定时器中断。为此,本研究在Windows平台上设计实现了精度为1 ms的定时器,并通过JNI技术将其与Java平台结合起来,从而实现了在框架内的实时时钟模型。

具体模型如图3所示。

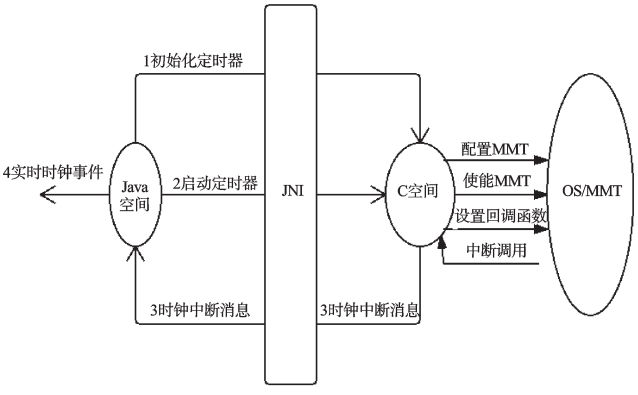

图3　Java平台的实时时钟实现

在该系统模型中,本研究采用了Java/C混合编程技术,其中核心是JNI。首先,Java的实时时钟模块在初始化时通过JNI调用响应的基于C的多媒体时钟初

始化函数进行初始化工作,最重要的是设置时钟周期,以及设置回调函数入口地址等;其次Java的实时时钟模块启动时,也要调用相应C函数启动多媒体定时器计数;启动完毕后,多媒体定时器会按照预置的时间间隔定期调用C设置的回调函数,而回调函数则通过JNI与Java虚拟机通讯,找到实时时钟模块并调用其时钟触发方法。此时,所有在Java平台中注册监听实时时钟消息的所有对象都会得到通知。

3.2 实时时钟开发过程简述

本研究所采用的工具是Sun公司创建的 Java Development Kit(JDK)版本1.6.0_22,以及微软公司的Visual C++ 6开发环境。本研究设计中的Java语言与C语言相互调用,这就要用到JNI。JNI是Java Native Interface的缩写,JNI是一种机制,有了它就可以在Java程序中调用其他本地代码,或者使本地代码调用Java层的代码。

在Java程序中,首先要编写带有native方法的类,要在该类中声明所调用的库名称System. LoadLibrary(String libname);在该类中只需要声明将要调用的本机方法,关键字为native,不需要具体实现,实现放在由C编写的DLL文件中实现。

本研究编写好带有本机方法的TimerManager类后,进行编译,生成对应的class文件。再用javah scc.TimerManager语句编译,就会生成对应的scc_TimerManager.h头文件。

之后就可运用该头文件中的函数原型:JNIEXPORT void JNICALL Java_scc_TimerManager_timer (JNIEnv *env,jobject obj)来编写所需要的DLL文件,来产生固定周期的时钟信号。

虽然Win平台下可视化开发工具如VC、Delphi、C++ Builder等都有专用的定时器控件Timer,而且使用很方便,可以实现一定的定时功能,但最小计时精度仅为55 ms,且定时器消息在多任务操作系统中的优先级很低,不能得到及时响应,往往不能满足实时控制环境下的应用。不过Microsoft公司在Win32 API函数库中已经为用户提供了一组用于高精度计时的底层函数,如果用户使用得当,计时精度可到1 ms。这个计时精度对于一般的实时系统控制完全可以满足要求。具体应用时,研究者可以通过调用timeSetEvent()函数,将需要周期性执行的任务定义在LpTimeProc回调函数中,从而完成所需处理的事件。需要注意的是,任务处理的时间不能大于所设置的时钟周期。定时器使用完毕后,应及时调用timeKillEvent()将之释放。

前面分析过Java层和JNI层(即C语言模块)应该可以互相交互,通过Java层中的native函数可以进入到JNI层,而JNI层的代码则通过JNIEnv操作Java层中函数。JNIEnv是一个与线程相关的代表JNI环境的结构体。JNIEnv实际上提供了一些JNI系统函数。通过这些系统函数可以调用java层中的函数或者操作jobect。GetObjectClass()方法可以通过对象jobect直接得到其所对应的类的定义。GetMethodID()方法可以得到所需要用到函数的id。而CallXXXMethod()函数则实现了java类中相应方法的调用。这样就完成了JNI层向java层的调用。要注意的是这里JNI层中调用的方法实际上是java中对象的成员函数,如果要调用static函数可以使用CallStaticXXXMethod()。这种机制实现了native层回调Java代码完成相应操作。

该系统实现中,在回调函数void CALLBACK TimeProc ()中调用CallXXXMethod (JNIEnv *env, jclass clazz)方法,从而调用java程序中的TimerManager类中的方法来触发一次时钟事件,产生周期为100 ms(可配置)的时钟。

3.3 实时任务调度模块的实现

实时时钟模块解决了实时性问题,但是大型控制应用往往都是多任务的,为实现多任务的实时运行,就需要利用实时任务调度模块。

本研究对实时任务的管理通过一种类似时间片的方式进行。首先,实时任务模块在初始化时需要在实时任务调度模块中注册实时任务信息,主要包括该实时任务的执行周期和任务序号;任务调度模块根据这些信息确定时钟触发的频率以及任务模块的调用顺序。在任务执行时,实时任务调度模块会创建优先级较高的线程来单独运行实时任务;而且其他非实时模块的任务仍然在其他的线程中运行,由于Java线程管理采用的是抢占式,可以保证实时模块的优先运行。

对于超时运行时间片的实时模块,目前的策略是监测并给出警告。实时任务模块的设计者应当保证实时模块的任务能够在最小的时间片长度内运行完毕。对于因为运算量巨大且无法优化从而需要较多计算时间的实时任务,只能通过增大时间片的方法解决,此时整个系统的控制周期也会随之增大。

控制周期、实时时钟周期和时间片之间的关系如图4所示。

从图4中可以看出,时间片是任务执行的最小单位,一个任务只能占用一个时间片,时间片是由任务调度模块在一个实时时钟周期内分配给各个实时任务模块的,其时长是可变的,取决于任务的数量和实时时钟周期的大小;实时时钟周期是确定的不可改变的,控制周期则由多个时钟周期组成,对于大部分实时控制系统而言,控制周期往往只有一个,因此可以

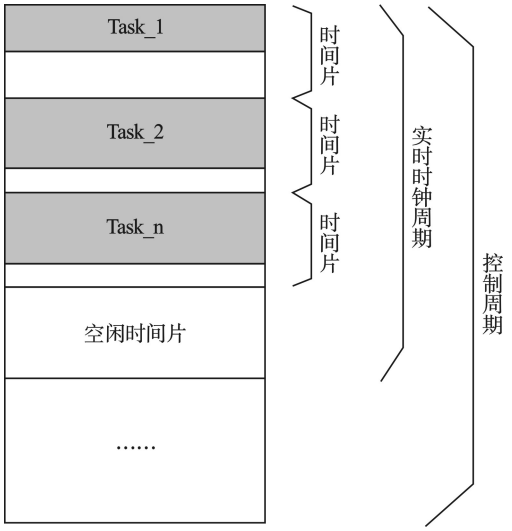

图4　控制周期，时钟周期和时间片之间的关系

将时钟周期设置为控制周期同样大小；如有多个实时控制对象，应将时钟周期设置为多个控制周期的最大公约数。

4　系统测试

为了测试系统的实时性，本研究只定义了一个实时模块和一个非实时模块来测试系统。该实时模块只有一个实时任务，其功能就是打印出运行时的系统时间，通过分析其间隔的时间精度与设计时的控制周期的误差来验证上述系统的实时性，加入非实时任务的目的是验证实时任务的实时性是否受非实时任务的影响。

测试结果如图5所示。从图5中可以看到，第1次与第2次调用时钟时系统时间间隔为98 ms，其余的时间间隔均为100 ms，与之前设定的时钟周期一致。可得出结论：前面设计的实时任务模块每隔100 ms就将模块中的实时性任务执行一次，可以满足整个系统的实时性要求。

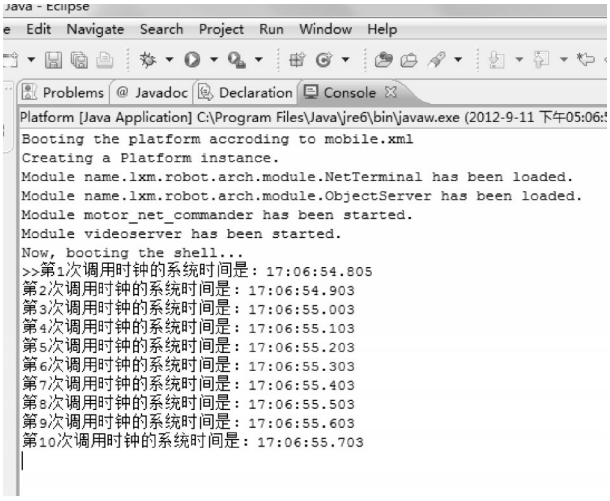

图5　测试结果

5　结束语

本研究在原基于组件的机器人控制软件框架的基础上，开发了实时任务模块，将原系统改造为支持实时性软件开发的软件框架，基于该框架，可以快速开发具有实时性功能要求的控制软件，重点解决了在Windows平台上实时时钟模块以及实时任务调度模块的实现方法。

目前，该软件框架正用于本课题组RoboCup小型组足球机器人控制策略软件的开发中，在后续工作中，笔者将重点对实时任务的调度算法进行研究，重点解决如何能够发现用户开发的实时任务中存在的问题，以及如何避免因某一实时模块的问题引起整个应用程序实时性变差的问题等。

参考文献(References)：

[1] 陈万米，张　冰，朱　明，等. 智能足球机器人系统[M]. 北京：清华大学出版社，2009.

[2] 李　策. 基于DSP和FPGA的小型足球机器人控制系统的研究[D]. 大连：大连理工大学电子与信息工程学院，2006.

[3] 徐璟业. F-180小型足球机器人的无线通信系统设计[J]. 电子技术应用，2004(5)：60-63.

[4] 张丛荣. 足球机器人运动控制系统的研究与完善[D]. 沈阳：东北大学信息学院，2006.

[5] LI Xiao-ming，JIN Yu-zhen，HU Xu-dong. An XML-driven component based Software Framework for Mobile Robotic Applications[C]//Proceedings of the 2nd IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications，MESA 2006. Beijing：[s.n.]，2006.

[6] BRUGALI D，SCANDURRA P. Component-based robotic engineering(part 1)[J]. **IEEE Robotics and Automation Magazine**，2009，16(4)：84-96.

[7] BRUGALI D，SCANDURRA P. Component-based robotic engineering(part 2)[J]. I**EEE Robotics and Automation Magazine**，2010，17(1)：100-112.

[8] INGHAM M，RAGNO R，WILLIAMS B. A Reactive Model-based Programming Language for Robotic Space Explorers[C]//Proceedings of ISAIRAS-01，2001. Montered：[s.n.]，2000.

[9] 胡海涛. 面向对象技术在实时系统中的研究与应用[D]. 西安：西北工业大学计算机学院，2003.

[10] AUSLANDER D M. What is mechatronics[J]. **IEEE/ASME Transactions on Mechatronics**，1996，1(1)：5-9.

[11] AUSLANDER D M，RIDGELY J R，RINGGENBERG J D. Control Software for Mechanical Systems：Object-oriented Design in a Real-time World[M]. 北京：清华大学出版社，2004.

[12] 李晓明，肖亮亮，孙琛琛. 基于VB的通用运动控制软件模型设计[J]. 机电工程，2009，26(12)：50-53.

[13] 赵翼翔，陈新度，陈　新. 全自动芯片粘片机点胶控制系统的设计与实现[J]. 机械与电子，2006(3)：33-35.

[编辑：罗向阳]