基于组件的开放结构数控系统*

陈友东 陈五一 王田苗
(北京航空航天大学机械工程学院 北京 100083)

摘要：提出并实现了一种模块化、可重构、可互换和可裁减的开放式数控系统结构，为快速定制和开发数控系统提供了一个软件结构和环境，该结构结合了通用 PC、RTLinux 和实时组件的优点。在分析现有组件标准不能应用于数控系统的原因的基础上，给出一种可用于数控系统的组件模型。介绍了由该模型实现的开放结构数控系统的基本组成成分，以及组成一个基本数控系统组件的功能、接口和图形表示。详述了该系统的虚拟模块系统和配置系统，虚拟模块系统主要实现数控系统模块间通信和模块的调度，配置系统主要实现数控系统的集成。给出了该组件的实现框架，并在 RTLinux 上实现了该结构的原型系统。

关键词：组件 开放结构 计算机数控
中图分类号：TP273

0 前言

所谓开放式数控系统就是能够在众多平台上运行，能与其他系统进行互操作，并能给用户提供统一风格界面的系统。根据这一定义，开放式数控系统必须是全模块化的体系结构[1]，它应具有以下几个主要特征：互操作性、可移植性、可裁剪性和互用性。因此开放式系统必须是中性卖主和组件集成的，中性卖主是指所设计的开放式系统基于事先制订的标准，该标准独立于单个专用卖主；组件集成则表示开放式系统具有高度可移植性和可扩展性。

组件是指系统中可替换的物理单元，该单元封装了模块的实现细节并提供了一组实现的接口[2]。组件是抽象的系统特征单元，它具有封装性和信息隐藏性，它的功能由它的接口定义，它可以是原子的，也可以是复合的，并且组件是可配置和共享的。组件独立于语言，它既可用面向对象编程语言和面向模式的语言实现，也可用非面向对象的过程语言实现。只要遵循组件的规范，每个软件开发商就可以使用自己方便的语言开发。基于组件的软件开发的目的是通过创建可以装配到软件系统的组件获得重用、经济和可靠性等优点。

基于组件的软件范型可以有效地应用于实时系统设计中。目前几个广泛使用的组件标准，如 Microsoft 公司的 COM/DCOM 标准、SUN Microsystems 公司的 JavaBean、OMG 组织的 CORBA 等，但是这些组件标准在实时系统中不适合使用，因为它们没有处理时间和服务预测性问题。组件 *A* 无法指定组件 *B* 在确定的时间内为其提供服务，甚至不能给定组件 *B* 为这个请求提供服务的时间上限；访问控制是另一个没有被传统接口规范涉及的组件接口特性，但是在临界系统(许多实时系统是临界的系统)中它是必须具有的接口特性，这样的系统需要限制授权客户访问关键服务的数量。在理想情况下，组件接口应包含每个客户访问的操作信息。使用这些组件的另一个缺点是在时间和空间上增加了系统开销，这些系统开销可能会对控制系统产生不利影响，因为一般控制系统的存储空间和处理器速度都有限。ORG 工作组提出了实时 CORBA 和嵌入式 CORBA 的规范。实时 CORBA 规范了资源(内存、进程、优先级、线程、协议和带宽)控制机制和在分布式系统中优先级的处理机制。嵌入式 CORBA 通过消除一些功能(如动态接口等)、使用操作系统提供的服务和专门的转换机制减少存储空间的使用量。主流的 ORB 厂商反对实时 ORB，而且硬实时 ORB 还有很长的路要走。有很多组织如 SUN 等都考虑对 JAVA 进行了实时嵌入式扩展，将 JavaBean 转变为适合实时组件的工具也许是第一步，但这仍是个长期目标。

在 RT-Linux 上实现了一种基于组件的开放结构数控系统，实现了模块化、可互换、可裁减的数控系统，提供了一个快速定制和开发数控系统的软件结构和环境。它支持译码、插补、人机界面和运动控制算法等的用户定制，实现系统良好的开放性。系统的组件模型是基于端口对象 PBO(Port-based

* 国家自然科学基金资助项目(60[illegible]刊初稿，20051215 收到修改稿

object)的模型扩展[3]。

1　RT-Linux

RT-Linux 是一个强实时操作系统，可用于机器人控制、数据采集、机床控制及其他对时间要求较高的设备控制系统。运行在 RT-Linux 上的实时进程作为轻量级线程执行，并运行在内核空间。RT-Linux 和 Linux 操作系统共存在一起，使用虚拟机层，使得 Linux 内核就象一个低优先级的实时进程一样在 RT-Linux 里运行。RT-Linux 和 Linux 的结合能提供实时功能和非实时功能，包括底层任务的创建、中断任务的安装、底层任务队列管理、中断任务的运行等[4]。

典型的 RT-Linux 任务由运行于核心态的实时任务和运行于用户态的非实时任务组成。RT-Linux 的实时任务作为内核模块执行，故它速度很快 。同时，实时任务不支持一般的 Linux 系统调用，它们必须通过进程间通信(Inter-process communication，IPC)与普通 Linux 任务交互后，由普通 Linux 任务执行系统调用。因此实时任务要求尽量简单，仅仅包含那些有强实时要求的处理模块。实时先进先出队列(First in first out，FIFO)和共享内存是实时模块与用户态模块通信的两种方式。用户态模块通过 FIFO 和共享内存访问数据，经过处理后把数据放到另一个 FIFO 和共享内存中供实时模块访问。实时任务的执行，可用实时中断处理例程和实时线程两种方式实现。图 1 显示了控制系统中 RT-Linux 组件间相互作用关系。

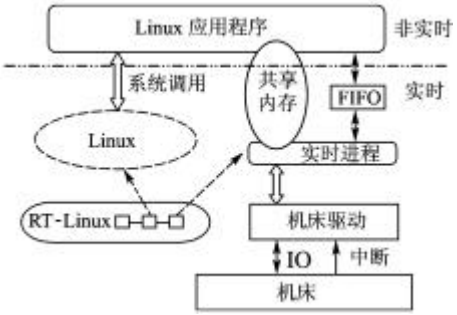


图 1　控制系统中 RT-Linux 组件间相互作用关系

2　开放结构数控系统组成

本数控系统主要由三部分组成：组件库、虚拟模块系统和配置系统。组件是该系统的基本软件模块，它封装了一系列互相关联的操作和服务，是一个可实际运行的软件模块。该系统的组件包括轴控制组件及粗插补组件等。组件间通过标准的应用程序接口(API)通信，隐藏了硬件细节并使相关组件以方便正确的方式通信。虚拟模块系统为数控系统提供了运行机制。它负责机床控制器和人机界面之间的联系，运行数控任务，负责任务的调度以及任务间数据的同步。配置系统包括模块配置器和文本编辑器。模块配置器是个监视任务模块装载、执行和重新配置的子系统，它是根据配置信息把各个模块连接在一起的连接器；文本编辑器是用来配置控制任务的。

2.1　组件库

组件如图 2 所示，它由端口和控制逻辑驱动器两部分组成。端口是任务模块间通信以及任务模块和外部环境联系的通道。控制逻辑驱动器将控制逻辑和功能定义分离开，它可以看作访问和修改任务模块内部控制逻辑的接口，每个涉及控制逻辑的任务模块内部都有一个这样的驱动器。任务模块的控制逻辑由状态表指定。任务模块运行时，控制逻辑驱动器根据状态表和外部事件产生指令，激活被控对象并执行操作。每个任务模块内部控制逻辑驱动器的运行框架都是一样的。

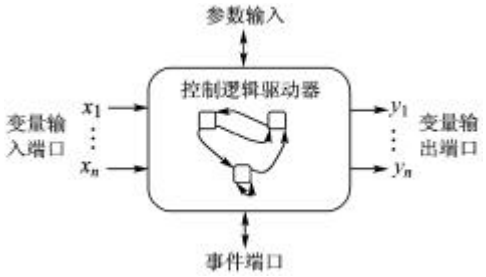


图 2　任务模块

组件给出了数控系统软件模块的基本结构和框架，系统开发人员在开发数控系统中完成特定功能的软件模块时，可以在继承组件所定义的内部状态和操作基础上，定义软件模块特定的状态和操作。

为实现数控系统的重用性和重构性，把这些组件存放在组件库中，以便在集成数控系统时从组件库中选择合适的组件。由于数控系统的组件不是很多，所以这里用文件系统实现组件库。组件库中存储组件的描述和代码、目标文件的名称。组件的目标文件存储在动态加载连接库中，供系统启动时动态加载。

2.2　虚拟模块系统

虚拟模块系统是为各个模块相互服务提供一个机制，之所以称它为虚拟模块系统是因为它所有的数据结构都是在系统运行以后才建立的，并在卸载时删除，而在磁盘上并没有存储这些数据结构。显然，如果只有虚拟模块系统，系统是无法工作的，

只有和实际的模块相结合才能工作，所以虚拟模块系统并不是一个真正的模块系统。

虚拟模块系统就是所有的组件以相同的方式连结在一个用来相互通信的结构性组件上。由于遵从统一的通信机制，完全实现了组件的通用化。在这种软件体系结构中，组件(主要是中间件)就像是硬件的“插件”，可以随意添加和删减，系统的灵活性和可靠性都大大提高了。虚拟模块系统模型如图 3 所示。

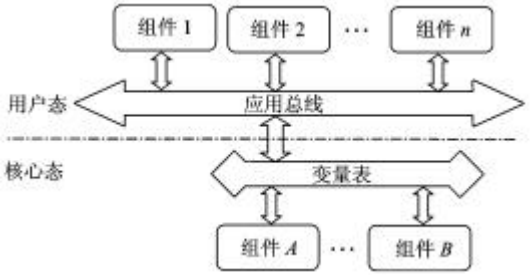


图 3 虚拟模块系统

运行在 RT-Linux 系统上的任务分为实时任务和非实时任务。实时任务运行在 RT-Linux 的内核空间，而非实时任务则运行在 RT-Linux 的用户空间，所以该模型分为两个部分——核心组件和应用组件。核心组件运行于操作系统的核心态，主要完成数控系统的实时任务，包括位置控制以及逻辑控制任务等。核心组件组成一个基本的数控系统，相当于一个运动控制器，在这里把它称为软运动控制器。这个简单的控制系统允许用户手工控制设备。用户通过操作界面得到设备当前的状态，也可以从操作界面通过全局共享内存向设备发送指令操纵它。

由于软运动控制器使用不是很方便并且不能满足需求，因此应用组件应运而生。应用组件是在软运动控制器的基础上开发的模块，它运行于操作系统的用户态，包括译码任务、粗插补等任务、人机控制任务、网络功能和诊断功能等。应用组件根据其时间特性又可分为非实时任务和软实时任务。

软运动控制器的数据通过共享内存和实时 FIFO 对应用组件开放。软运动控制器和应用组件通过软总线集成。软运动控制器里的各个组件是通过全局/局部共享内存表集成在一起的。

虚拟模块系统为运行于其上的数控系统的各个功能组件提供一种任务协调与同步机制，因此虚拟模块系统由任务调度器、通讯系统组成。任务调度器控制着任务对处理机的使用并监控各任务的状态，根据调度策略和调度算法改变任务的状态，或让其运行、或让其等待，这里采用的是 RT-Linux 自带的调度器—基于优先级可抢占调度器。通信系统与系统的软硬件平台有关，它提供开放式数控系统所需的通信机制、通信协议与接口。

2.3 配置系统

开放式数控系统配置系统就是利用组装的方法把已有的任务模块组装在一起，从而构造新的可用的数控系统。它要处理数控系统的拓扑关系、在数控系统启动阶段任务初始化模块的参数、组织配置数据的读取、创建全局变量表、装载任务模块、建立任务模块的运行实例、建造任务模块间的通信连接和启动任务模块。配置系统只涉及软件的配置，其组成如图 4 所示，它由文档编辑器、任务模块库、组装系统和拓扑结构组成。

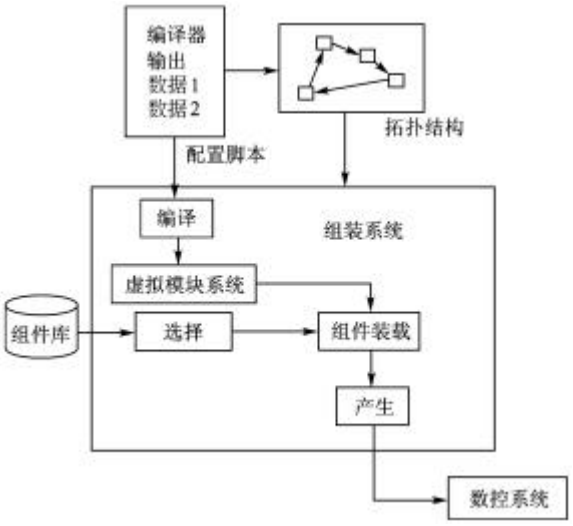


图 4 配置系统组成

(1) 文本编辑器。文本编辑器主要对配置文件进行编辑，它可以是任何一个文本编辑器，也可以是个图形编辑器。系统设计人员通过创建任务模块的数据流图配置控制任务。它运行在非实时的计算机上，可以通过网络把配置信息传递给组装系统进行系统的合成。

(2) 任务模块库。任务模块库是一系列具有标准化接口、良好说明文档的任务模块，这些任务模块中，实现相同功能的任务模块可以互换。任务模块一般保存在操作系统的动态连接库中，这样可以动态加载任务模块。

(3) 拓扑结构。拓扑结构是指数控系统的软件拓扑结构，由配置文件描述。它决定了任务模块间的执行顺序和相互关系。不同的数控系统，尽管其所适用的机床类型不同，但它们的运行过程是相似的，都是对零件加工程序的解释、处理、插补和位置控制这样一个循环往复的过程，这就决定了它们的拓扑结构相似性，它不会因功能的实现形式、用户需求和加工条件的不同而发生变化。

(4) 组装系统。组装系统的功能是根据配置文件存储的配置信息，从组件库中选择组件，创建任务模块的运行实例，参数化任务模块以及内部对象，生成组件间的通信连接，组织组件间的信息流，形成新的数控系统。组装系统的主要功能有两个：编译和装载。编译就是扫描配置文件，识别模块的名字、模块的输入输出数据端口、模块的类型、模块的运行周期、模块的优先级等。装载就是根据模块的输入输出数据创建全局变量表、根据模块名称从动态库中装载模块，根据模块的运行周期和优先级初始化模块，建立模块和全局变量表间连接，启动模块。建立模块和全局变量表之间的连接也就是建立了模块间的通信连接，建立了控制系统的软件拓扑结构，当然它也进行组件接口正确性和一致性检查。

一般配置过程是：用户开发出任务模块，利用配置编辑器离线编写配置文件(可以是文本编辑器，也可以是图形编辑器)，在控制系统启动时，配置系统根据配置文件动态地完成控制系统的配置。该系统提供一个缺省的数控系统，供用户使用。

3　数控系统组件

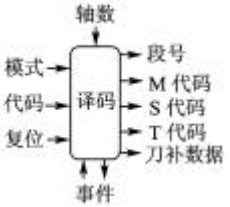
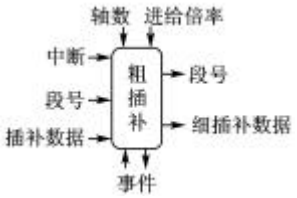
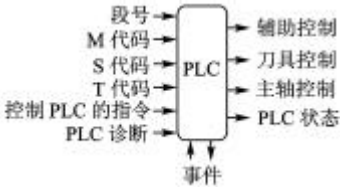
开放式数控系统只有在被划分为小的功能模块、各个功能模块的接口事先指定的情况下，才能够被实现。模块化是识别开放式系统的关键。确定模块复杂性时，显然在开放程度和集成代价上有个平衡。小的模块提供更高层次的开放性和更多的选择，但是增加了系统的复杂性和系统集成的难度。另外，这样细粒度的系统对系统资源要求更高，也就是消耗系统的资源更多，这样可能降低整个系统的实时性能。

将数控系统中的各级功能模块自上而下进行细分，得到“系统—任务—组件”的树状层次结构。组件是该系统中最小的组成单位，它是个任务模块，同时也是可复用的二进制文件。功能相关和周期相同的组件集成为一个任务，每个任务完成一定的功能，任务可以嵌套，即可在一个任务中包含一个或几个下级任务。一系列相关的任务组成某种类型的数控系统，一个组件或任务在一个系统中被实例化为一个组件对象或任务对象。一个系统可以包含一个或多个组件对象或任务对象。

由于研究的重点是在系统体系结构的开放性上，是以研究为目的的原型系统，不追求数控功能上的先进性与完备性。因此该研究提供了一个基本的数控系统，该系统只考虑了译码器、刀具补偿、粗插补器、细插补器、轴控制、PLC 组件和队列等组件。

下表给出了组成数控系统的组件库，图 5 给出了数控系统的集成框架。该系统从人机界面获得加

表　数控系统基本组件

类别	图形表示	功能概述
人机界面		人和机床控制器联系的桥梁。负责管理命令和控制，同时监视系统的状态
译码器		负责加工程序的词法检查、语法的检测和代码的分离，并进行相关处理，如英制转换等任务。周期性任务，运行在 Linux 用户空间
刀具补偿		完成刀具补偿、速度计算以及辅助功能等任务。周期性任务，和译码器的周期相同。运行在 Linux 用户空间
粗插补器		把给定的运动轨迹按照一定的周期插补为若干段。它依赖于机床的具体结构形式，是个周期性任务，它运行在 linux 用户空间中
细插补器		对粗插补器输出的各坐标轴进给量进一步细化。周期性任务，运行在 RT-Linux 空间。
轴控制器		负责控制单轴的运动，可能执行开环、闭环或逻辑控制。周期性任务，运行在 RT-Linux 空间
PLC		PLC 主要完成机床的辅助功能和机床逻辑控制。运行在 RT-Linux 空间
队列		为运行周期不同的任务提供一个同步机制
事件服务器		监视各个组件的运行情况，对事件作出相应的处理，发出相应的消息

工代码和指令，由译码器对加工代码译码成两部分数据，一部分是位置控制信息，经刀具补偿器，粗插补等模块控制机床各个轴的运动；另一部分是 PLC 模块接收的开关量信号。

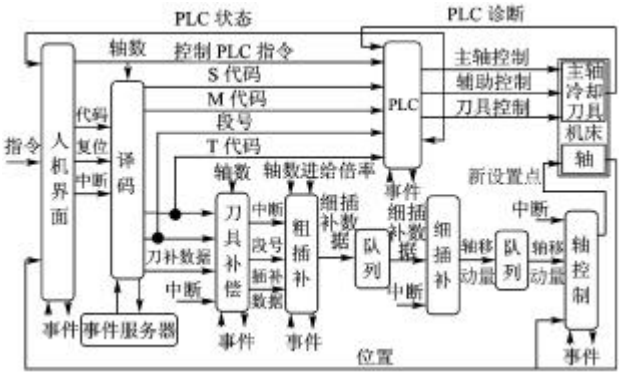


图 5 数控系统的集成结构

4 数控系统组件的开发

任务模块设计目标是给出设计任务模块一个通用的框架。该框架规定了任务模块的结构。它定义了任务模块整体结构、对象、各部分的主要责任、各部分的协作以及控制流程。框架预定义了这些设计参数，以便模块设计者或实现者能集中精力于模块本身的特定细节。框架使设计者可以复用任务模块的主体，只简单的定义模块专用的代码即可，而不涉及任务模块主体的任何细节。下面给出了组件的实现框架。

state= 未创建
while state = 未创建 暂停或等待 do
case state of
未创建：
　拷贝配置信息；
　初始化；
　和虚拟模块系统建立连接；
　进入暂停状态；
end case;
暂停：
　接收消息；
　如果消息是 Kill, 则退出进入未创建状态；
　如果消息是 On, 则启动该任务，和虚拟模块系统进行通信；如果是其他状态则报错并退出
end case;
等待：
　接收消息；
　如果消息是 Kill, 则退出进入未创建状态；
　如果消息是 Off，则暂停该任务，和虚拟模块系统进行通信，任务进入暂停状态；如果消息是 Run，则执行该任务，和虚拟模块进行通信，任务进入暂停状态；如果是其他状态则报错并退出
end case;
其他状态:
　退出；
end case
end while

这里采用基于对象的技术进行设计和开发，用 C 语言进行开发。开发的 CNC 组件放置在组件库中。这些组件是控制工程师根据需要开发的。一般应用工程师在使用这些组件构建一个新的系统时，他们不需要知道这些功能是如何实现的，他们只需为他们的系统选择恰当的组件即可，然后根据组件的接口说明使用组件。

在 RT-Linux 上实现了其原型系统。该系统运行于 RedHat9.0+RT-Linux v3.1 的操作系统上。该数控系统主要由译码器、数据处理、粗插补器、细插补器、轴控制、队列、人机界面及事件服务器等模块通过配置文件配置而成。

5 结论

针对开发开放结构数控系统的需求，研究出一种基于组件的系统框架，并以此为基础在 RT-Linux 上实现了该系统框架。该框架为快速定制和开发数控系统提供了一个软件结构和环境。通过实时组件的模型设计的数控系统，既实现了系统开放性，也为开发数控系统提供了一个快速的途径。在此基础上，根据双内核实时系统(如 RT-Linux)的特点，把组件分为核心组件和应用组件，并把核心组件归类为软运动控制器，为在双内核实时系统上开发控制系统提供了一种模式。

参 考 文 献

[1] PRITCHOW G, DANIEL C, JUNGHANS G, et al. Open system controllers a challenge for the future of the machine tool industry[J]. Annals of the CIRP 1993, 42(1)：449-452.

[2] RUMBAUGH J, JACOBSON I, Boock G. The unified modeling language reference manual[M]. Addison Wesley Longman, Inc. , 1999.

[3] STEWART D B. Designing software component for real-time applications [C]//1999 Embedded Systems Conference, San Jose, CA, 1999：408-423.

(下转第 198 页)