共享变量技术在 CompactRIO 中的应用

Shared Variables Used in CompactRIO

何芝霞　黄　昶　倪瑞萍　(华东师范大学 信息学院电子科学系 上海 200241)

摘　要：提出了一种基于 Lab VIEW 网络共享变量技术的分布式主从机之间的通信方式,实现了 PC 和实时目标 NI CompactRIO 之间一对多的访问控制模式,极大地简化了分布式系统间的网络通信编程。

关键词：网络共享变量 实时目标 动态绑定 分布式系统

美国国家仪器公司(National Instruments,**简称** NI)**最新推出的** CompactRIO－9012 **是具有极高实时性的嵌入式控制和采集平台,适用于可靠的独立式或分布式应用系统。**CompactRIO **通过** LabVIEW RT **模块编程,它作为下位机没有提供应用程序的用户界面,因此** CompactRIO **产生的数据必须通过网络通信传输到主机上显示或保存。**LabVIEW **为创建分布式应用提供了多种多样的技术接口,典型的网络通信方式包括** TCP、UDP、VI Server、**网络共享变量等。**Lab VIEW RT **模块提供的共享变量技术用于网络通信开发最为简单,为简化分布式系统间的网络通信编程迈出了重大一步。**

1　共享变量技术

共享变量是在 LabVIEW 8 **中引入的一种新技术,它使得在** LabVIEW **应用中传递数据更容易实现。图** 1 **描述了基于网络共享变量的** RT **目标** CompactRIO **和主机之间的数据通信。**

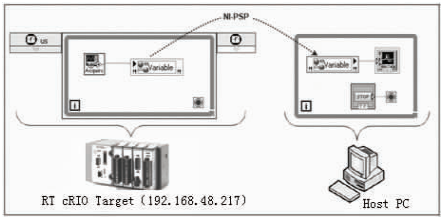

图 1　应用共享变量技术的数据通信

共享变量有 3 **种类型**:Single－Process、Network－Published **和** Time－Triggered。Single－Process **类型的共享变量只能用于本地** VI **之间或循环之间传递数据。**Time－Triggered **类型的共享变量应用于实时通信。**Network－Published **类型的共享变量用于网络上不同节点之间传递数据。与** Lab VIEW **中其他现有的数据共享的方法如** UDP/TCP、LabVIEW **队列以及实时** FIFO **不同,**Network－Published **类型的共享变量通常在编辑时使用属性对话框进行设置,而不需要在应用中包括配置代码。**

共享变量引擎(Shared Variable Engine,SVE)**是一个使网络发布的共享变量能够通过网络传送数据的软件框架,共享变量底层的网络通信、缓存器管理等均是由共享变量引擎** SVE **实现的。为了使用网络共享变量,**SVE **必须运行在分布式系统中的至少一个节点上。在使用网络共享变量之前,必须将网络共享变量发布到** SVE **上,**SVE **为该网络共享变量分布内存,并将其在网络上发布,从而使得网络上的任何节点都能够读写该共享变量。网络共享变量的通信是通过** NI－RSP **协议通信的,**NI－RSP **协议在继承** UDP **的无状态和非连接导向特点的同时,还增加了一层功能实现了数据传输的可靠性。**

2　共享变量技术在 RT 中的应用

在分布式系统中,通常需要上位机同时远程监控多台下位机的运行情况。例如,在某一测试多种类型开关性能的分布式系统中,配备了多个 CompactRIO,**每个** CompactRIO **实现一种类型开关的测试。要达到同时对多种类型开关进行测试,就需要实现主机与多个** CompactRIO **之间的数据通信。下面以一个远程监视多个** CompactRIO **上的变量为例介绍共享变量技术**

在 RT 目标中的应用。图 2 给出了在一个局域网中,用户在上位机通过切换不同的 CompactRIO 读取它们各自的共享变量值的操作界面,波形图反应了共享变量值随时间的变化。

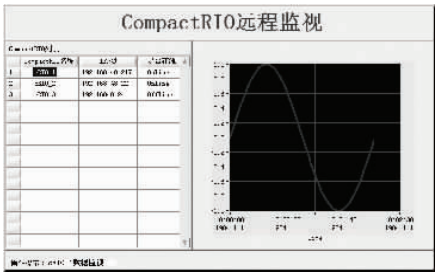

图 2　多个 RT 目标的监视界面

2.1 创建共享变量

为实现将 RT 目标 CompactRIO 产生的数据通过网络传输到远程上位机上显示,必须首先要在每个 CompactRIO 上创建自己的共享变量库。共享变量库包括两个变量: CompactRIO 上的时间(RIODateTime,数据类型为 string)和产生的波形数据(Data,数据类型为 double);同时,上位机要访问局域网中多个 RT 目标,也需要在本地对应每一个 RT 目标创建一个共享变量库,其共享变量名和数据类型与 RT 目标上的完全一致。

2.2 部署共享变量

部署共享变量就是在网络节点上安装 SVE,可以在 Project Explore 的右键菜单中手动部署共享变量。但是在可执行程序中不能以这种方式部署共享变量,必须通过编程来实现,分为两种情况:一是在 Windows 下部署共享变量时可以利用数据监控模块(DSC)中 Deploy Library. vi;二是部署共享变量到 RT 目标,可以通过设定方法结点 Deploy Library 的 Liabrary Path 和 RT 目标 IP 地址来实现。共享变量部署完成之后可以通过工具 Variable Manager 查看本地和 RT 目标上部署网络共享变量的状态。

2.3 动态绑定共享变量

在创建共享变量的属性配置对话框中可以手动设定共享变量绑定的数据源,即将上位机上的共享变量与局域网中 RT 目标上已经部署了的共享变量绑定,绑定路径一旦确定之后上位机的共享变量就只能读取其绑定路径确定的数据源(RT 目标)的共享变量数据;另外一旦 RT 目标的 IP 被修改之后,也需要重新编译程序,这种绑定方式只适用于一对一的通信方式。在一对多、多对多的分布式系统中,通常需要通过一个上位机访问多个下位机,这时可以通过编程实现动态绑定上、下位机的共享变量,从而灵活改变与上位机进行通信的 RT 目标。

动态绑定共享变量,可以利用 DSC 模块的属性结点 SharedVariableIO,选择 Network > > URL,将要绑定的远程变量的 URL(格式: \0. 0. 0. 0 \Variable Library \ Shared Variable)赋给这个属性,并在这个属性结点上增加另一个元素 Use Binding,同时将其赋值为 True,这样就实现了本地和远程 RT 目标的相应共享变量绑定。

由于采用了动态绑定共享变量机制,当在上位机界面上切换 RT 目标时,本地和远程 RT 目标的共享变量绑定也随之发生改变,这样上位机就可以读到当前下位机的数据,从而实现同一个上位机程序对多台下位机进行监控的目的。

2.4 共享变量使用的注意事项

2.4.1 共享变量 Quality 检测

在分布式系统中,本地和远程共享变量绑定成功是程序正常运行的关键,通常造成绑定失败的原因有两个:(1) 绑定路径为空;(2)有绑定路径,但本地共享变量的 Quality 不是 Good。我们可以通过工具 Variable Manager 查看共享变量绑定失败的原因,将要查看的共享变量库的所有变量添加到 WatchList 中,查看其绑定路径和 Quality。若绑定路径为空,一般是由于绑定前的数据流中出现了 Error;对于解决 Quality 不为 Good 的方法是在绑定之后去读变量的 Quality 直至其变为 Good,这样做可以保证绑定成功。但是如果一直读取不到 Quality 为 Good,则非常耗费时间,导致效率低下。实际中可以通过设定检查 Quality 合适的次数达到一定的平衡。图 3 所示的框图程序采用 DataSocket 方式读取共享变量的 Quality,同时设置了读取次数为 11 次,如果超过 11 次后仍然读取不到 Quality 为 Good,则退出循环给出 error 信息。

实际应用中如果需要确保读取到的共享变量数据正确,还可以在读取共享变量数据的同时,检测共享变量的 Quality。读写共享变量数据一般可以通过共享变量引用或者 DataSocket 读写函数,采用 DataSocket 的方法要复杂一些,需要 Open、Read/Write、Close 三个步

骤,所以可以采用直接读写共享变量引用的方法。而检测共享变量 Quality,目前须采用图 3 中介绍的 DataSocket 方式,于是可以结合以上两种方法同时读取共享变量的数据和 Quality。图 4 所示程序框图中,上方的 While 循环框图读取共享变量数据,同时下方的 While 循环框图读取共享变量的 Quality 是否为 Good 的情况,其中的子 VI 就是图3 的框图程序。如果在读取数据的过程中,检测到共享变量的 Quality 不为 Good,则给出提示让用户做适当的处理(如让用户检查网络情况等)。

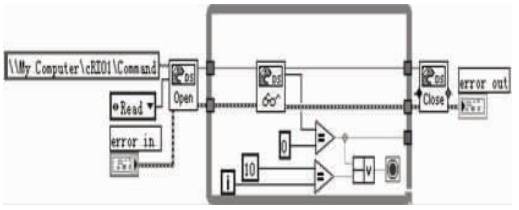

图 3　读共享变量 Quality 的程序框图

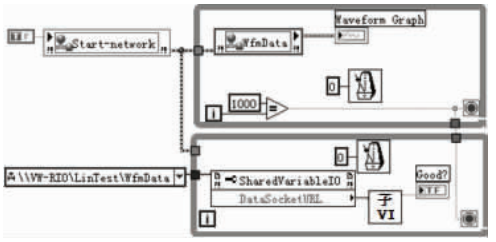

图 4　读共享变量数据的同时检测 Quality

2.4.2 共享变量传输效率

共享变量的数据类型可以是任何 LabVIEW 支持的数据类型,同时它还支持自定义数据类型。在共享变量属性对话框的 Data Type 下拉列表中选择 From Custom Control,可以将一个自定义数据类型(比如簇类型)与共享变量连接。测试结果表明,把多个数据打包在构成一个自定义数据类型的共享变量传输的效率要比分开由多个共享变量传输的效率高。图 5 显示了共享变量分散绑定和簇绑定的效率测试结果。从图中可以看出,当数据量只有一个时,分散绑定和簇绑定的效率相差不大。但是当有多个数据量时,簇绑定的效率明显高于分散绑定。如将四个数据量打包到一个簇类型的变量后,绑定时间约为 2.5 秒,而将四个数据量分散进行绑定,则需要 9 秒的时间。不同测试环境下,由于网络通信质量不同,得到的具体绑定时间可能不一样,但是绑定时间的比例关系基本与图 5 的曲线图一致。从图 5 可以看出,簇绑定的效率明显高于分散绑定,因此在实际开发中,应该尽量使用簇绑定来提高共享变量的传输效率。以上测试结果是在上位机为 Windows XP 操作系统 + LabVIEW 8.2,下位机为 CompactRIO – 9012 的运行环境下,对 bool 型的共享变量进行测试得到的。

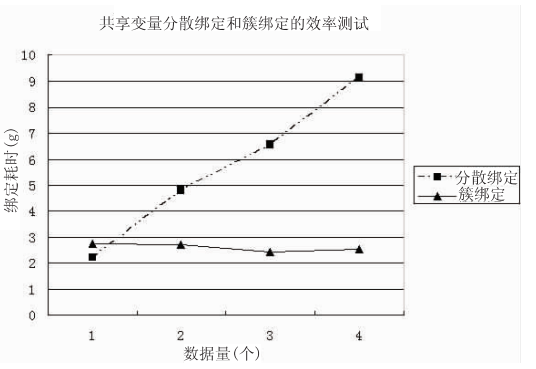

图 5　共享变量分散和簇绑定的效率测试结果

3　结束语

本文介绍了一种利用 LabVIEW RT 模块共享变量技术,在分布式应用系统的上位机和下位机之间进行网络数据通信,从而实现了 PC 和实时目标 NI CompactRIO 之间一对多的动态访问和控制。利用 LabVIEW 的共享变量技术,结合 NI 公司的 CompactRIO 平台,可以高效、便捷地开发出满足各种应用需求的分布式系统,极大地缩短了开发周期,提高了开发效率。

参考文献

1 陈锡辉,张银鸿. LabVIEW 8.20 程序设计从入门到精通. 北京:清华大学出版社,2007.

2 使用 LabVIEW 共享变量(技术指南). (2008 – 1 – 25) [2008 – 4 – 27], http://zone.ni.com/devzone/cda/tut/id167.