文章编号:1671-4598(2019)10-0154-05 DOI:10.16526/j.cnki.11—4762/tp.2019.10.031 中图分类号:TJ06 文献标识码:A

基于 NI PSP 协议的分布式测试系统软件设计

高 涛[1]，常 娟[2]，高艳军[1]，段海彬[1]
(1. 西安翔迅科技有限责任公司，西安 710068；2. 空军工程大学 基础部，西安 710051)

摘要：分布式测试系统广泛应用于工业、农业、军事和医疗等众多领域；随着计算机网络技术和分布式处理技术的发展，对分布式测试系统的软件开发提出了更高的要求，既要能满足众多节点的数据传输，同时也要保证在传输过程中数据的准确性；为了简化分布式测试软件的开发，保证数据传输的可靠性，提高软件复用性和开发效率，基于 NI Publish—Subscribe Protocol (NI PSP)，采用共享网络变量技术设计了可满足多节点大数据量传输要求的分布式测试系统软件；经过实际工程验证，软件实现简单，运行稳定，满足设计要求。

关键词：分布式系统；NI PSP 协议；共享网络变量；软件设计

Software Design of Distributed Testing System Based on NI PSP Protocol

Gao Tao[1], Chang Juan[2], Gao Yanjun[1], Duan Haibin[1]
(1. Xi'an Xiangxun Technology Co., Ltd, Xi'an 710068, China;
2. Science Institute, Air Force Engineering University, Xi'an 710051, China)

Abstract: Distributed test system is widely used in industry, agriculture, military and medical fields. With the development of computer network technology and distributed processing technology, higher requirements have been put forward for the software development of distributed test system. It should not only satisfy the data transmission of many nodes, but also ensure the accuracy of data in the transmission process. In order to simplify the development of distributed test software, ensure the reliability of data transmission, and improve the reusability and efficiency of software development, based on NI publish—subscribe protocol (NI PSP), a distributed test system software which can meet the requirements of large data transmission is designed by using shared network variable technology. Through the actual engineering verification, the software is simple to operate and stable in operation, which meets the design requirements.

Keywords: distributed test system; NI PSP; shared network variables; software design

0 引言

分布式测试系统是计算机网络技术的产物，是一种通过局域网或 Internet，把分布于不同地点、独立完成特定测试功能的计算机连接起来，达到测试资源共享、分散操作、集中管理、协同工作等目的网络测试系统[1]。测试计算机(下位机)一般位于测试现场的被测对象附近，可独立完成数据采集和预处理任务，并通过以太网将数据传输给控制计算机(上位机)[2]。上位机接收各下位机传送来的数据进行储存和分析处理，以图形化的方式显示数据，生成测试报表供用户查看。

按照通常的分布式测试软件设计方法，需要在下位机软件中实现数据采集和组包操作，然后通过以太网完成数据包的发送；在上位机软件中通过 Winsock API 函数接收数据包，并按照约定的数据包格式完成数据的拆包和提取显示[3]。进行软件开发时，为保证数据传输的可靠性，需要考虑数据包校验、错误重发、FIFO 缓存队列等流程的设计，开发难度大且容易出错，对软件开发人员有比较高的要求。此外，当系统中有多个下位机同时工作时，大量测试数据通过以太网传输到上位机的过程中，如何保证数据的一致性将成为一个难点[4]。

为了简化这种应用的软件开发，NI 公司提供了网络变量库(Network Variable Library)。该函数库基于 NI PSP (NI Publish—Subscribe Protocol) 协议，通过网络变量对底层网络通讯协议(TCP/IP 和 DDE)进行了抽象[5]。用户在 NI 公司的 LabVIEW、LabWindows/CVI 和 NI Measurement Studio 开发环境中使用网络变量库提供的 API 函数，可以在不影响系统性能的情况下，在多个系统或多个应用程序之间灵活的传递数据。

1 分布式测试系统设计

某分布式测试系统采用星型网络结构，多台远程数据

收稿日期:2019-03-04； **修回日期**:2019-03-23。

作者简介:高 涛(1981-)，男，山东宁津人，硕士，高级工程师，主要从事航电系统自动测试技术方向的研究。

采集终端 RTU（Remote Terminal Unit）作为下位机布置在测试现场，每台 RTU 有 50 个采集通道，每一台 RTU 均通过以太网与远离测试现场的测控计算机（上位机）相连。这种星型结构的优点是网络控制容易、便于扩充、可靠性好[6]。

开始测试时，上位机通过以太网发送命令控制 RTU 进行数据采集，RTU 以每个通道 10k/s 的采样率进行数据采集，并通过以太网将数据传输到上位机；上位机接收 RTU 传输的测试数据，对不同 RTU 的数据分别进行解析和显示，供用户查看。RTU 的 AD 采样精度为 16Bit，通过计算可以得出每台 RTU 在 1 秒内采集的数据量为 10 k×50 ch×2 Byte=1 000 kByte。

操作人员在远离测试现场的地点通过操作测控计算机查看系统中各个 RTU 采集的测试数据，支持多个测控计算机共享数据，分布式系统的组成如图 1 所示。

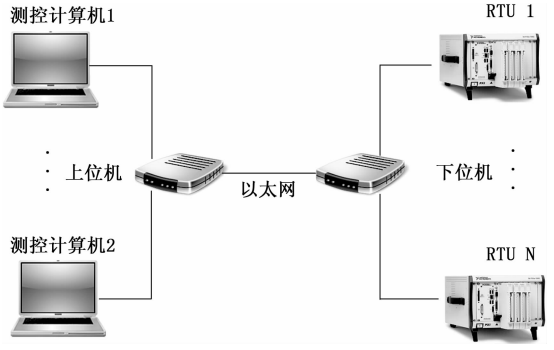

图 1　分布式系统组成框图

2　软件设计与实现

该系统中的 RTU 和测控计算机均运行 Windows 操作系统，各个 RTU 运行相同的下位机软件，通过配置文件给不同的 RTU 分配不同的 IP 地址，主要功能模块包括数据采集、组包、共享变量数据发布。测控计算机运行上位机软件，同样通过配置文件中的 IP 地址来控制不同的 RTU 协同工作，主要功能模块包括共享变量数据订阅、数据解包、处理、显示和存储。上下位机软件均使用 NI LabWindows/CVI 环境进行开发。

下位机软件作为测试数据的发布者，上位机软件作为数据的订阅者。在下位机和上位机之间使用 NI PSP 协议进行数据传输，首先需要创建共享网络变量并进行配置。

共享网络变量被部署到一个用于在网络上托管该变量的共享变量引擎 SVE（Shared Variable Engine）。SVE 是共享网络变量的服务器，所有对共享变量的应用（读或写）都是客户端。网络变量首先向共享变量引擎服务器发送数据，然后该服务器将这些数据发布给网络上的所有节点。因为数据是采用“发布－订阅”消息结构来传输的，所以客户端节点无需编写复杂的代码就可以访问网络变量数据[7]。

共享变量的创建和配置可以采用以下两种方式：一是

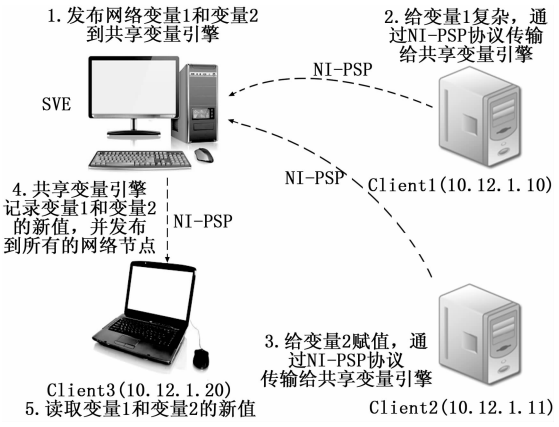

图 2　共享变量传输过程

使用 NI 提供的 Distributed System Manager 软件，在软件中以图形化的方式创建共享网络变量，另一种是使用 Network Variable Library 库函数，通过编程的方式创建共享网络变量[8]。

a）显式创建共享网络变量

图 3　分布式系统管理器软件界面

安装完 LabWindows/CVI 开发环境后，系统中安装了“Distributed System Manager”软件工具，软件路径如下：“开始 \ 所有程序 \ National Instruments \ Distributed System Manager”，软件界面如图 3 所示。

使用 Distributed System Manager 软件创建共享变量的步骤如下：

1）选中“localhost”项，通过鼠标右键菜单的“添加进程”创建共享变量进程“my _ process”；

2）选中刚创建的进程，通过鼠标右键菜单“添加变量”创建共享变量“my _ variable”，并设置变量类型和网络，同一个进程下可创建多个变量；

3）选中刚创建的进程，通过鼠标右键菜单进行“启动进程”、“停止进程”和“删除进程”等操作。

当网络变量创建好后，网络中的其它计算机就可以通过网络变量路径来共享数据。共享变量的网络路径包括计算机名称、进程名称以及共享变量名。例如：“\ 192.168.1.100 \ my _ process \ my _ variable”，其中“192.168.1.100”是数据发布计算机的 IP 地址，“my _ process”是网络变量进程，“my _ variable”是网络变

量名[9]。

b）编程创建共享变量

LabWindows/CVI 开发环境提供了 Network Variable Library（网络变量函数库）来方便软件开发人员在程序中对共享变量进行操作。Network Variable Library 库函数结构如图 4 所示，主要功能包括共享变量配置、数据包处理、变量发布和订阅操作等。

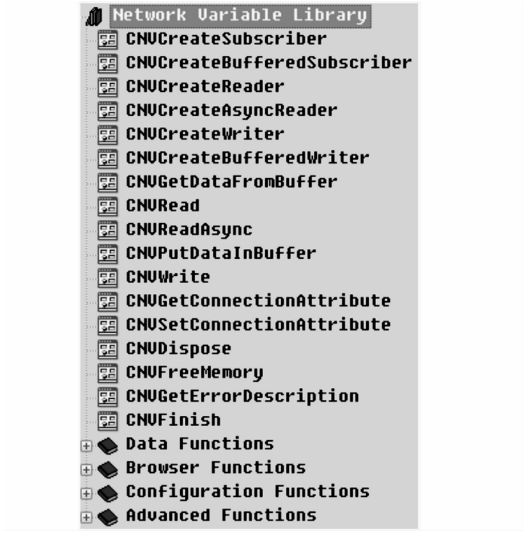

图 4 Network Variable Library 库函数结构

通过编程方式使用共享网络变量更为灵活，下面结合软件设计详细描述网络变量的编程使用方法。

2.1 下位机软件设计

该分布式系统的下位机负责采集测试现场的数据，是数据的发布端。下位机软件的流程设计如图 5 所示。

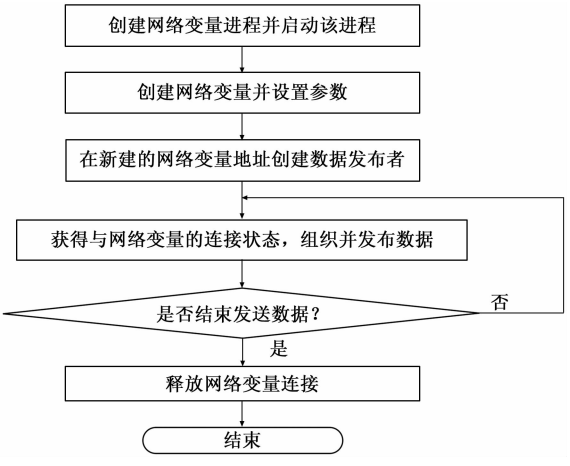

图 5 下位机软件设计流程图

第一步，首先创建网络变量进程，通过调用函数库中的 CNVNewProcess（" my _ process"）函数新建名为" my _ process" 的网络变量进程，并使用 CNVStartProcess（" my _ process"）函数来启动该进程。

第二步，创建网络变量，通过调用网络变量函数库中的 CNVNewVariable（" my _ variable"）函数新建名为" my _ variable" 的网络变量，使用 CNVSetVariableAttribute（）函数可以对该网络变量的属性进行设置，比如将 CNVVariableSingleWriterAttribute 属性设置为 1，说明该网络变量只有 1 个写入者；CNVVariableServerBufferMaxItemsAttribute 属性可以设置在服务器端能容纳数据项的数量。

第三步，使用网络变量函数库中的 CNVCreateWriter（）函数在刚才新建的网络变量地址：“\ IP \ my _ process \ my _ variable”创建数据发布者；当需要更新的数据量较大时，可以使用 CNVCreateBufferedWriter（）函数来创建发送缓存区，提高数据的发送效率。创建完成后可以使用 CNVGetConnectionAttribute（）函数获得与共享网络变量的连接状态。

第四步，发布数据，通过共享网络变量传输的数据类型可以是单个变量、字符串，也可以是一个数据结构。

共享网络变量的数据类型如表 1 所示，用户可以根据需要设置共享变量的数据类型。软件在通过共享网络变量进行数据发布和读取时，均是按已设定的数据类型进行操作，不用考虑在传输中的数据转换问题。

表 1 共享网络变量数据类型

数值类型	变量类型
无符号整型(1 字节)	CNVInt8
无符号整型(2 字节)	CNVInt16
无符号整型(4 字节)	CNVInt32
无符号整型(8 字节)	CNVInt64
有符号整型(1 字节)	CNVUInt8
有符号整型(2 字节)	CNVUInt16
有符号整型(4 字节)	CNVUInt32
有符号整型(8 字节)	CNVUInt64
单精度浮点型	CNVSingle
双精度浮点型	CNVDouble
字符串	CNVString
结构体	CNVStruct

当进行单一数据类型的数据传输时，通过使用 CNVCreateScalarDataValue（）函数创建不同类型的数据，再通过 CNVWrite（）函数发布到共享共享网络变量。

当进行复杂数据的传输时，需要将发送的数据组包为结构体进行发送。将数据组包为结构体的方法如下：

1）首先根据需要定义网络变量数据，数据类型为：CNVData。使用 CNVCreateArrayDataValue（）函数创建数组型的数据；使用 CNVCreateScalarDataValue（）函数创建数值型的数据；最后使用 CNVCreateStructDataValue（）函数创建包含若干个网络变量数据的结构体，示例代码如下所示；

```
CNVData data, fields[3]; //3 个变量数据
Check(CNVCreateArrayDataValue (&fields[0], CNVInt16, &dataArray, 1, dims)); //数据数组
```

Check(CNVCreateScalarDataValue (&fields[1], CNVUInt16, 0)); //通道号

Check(CNVCreateScalarDataValue (&fields[2], CNVUInt32, 0)); //包序号

Check (CNVCreateStructDataValue (&data, fields, FIELDSIZE));

2）接下来将数据填充到刚才创建的结构体中，示例代码如下：

Check (CNVSetArrayDataValue (fields[0], CNVInt16, DataBuf, 1, dims));

Check(CNVSetScalarDataValue (fields[1], CNVUInt16, i));

Check (CNVSetScalarDataValue (fields[2], CNVUInt32, pkgIndex++));

Check(CNVSetStructDataValue (data, fields, FIELDSIZE));

3）最后使用 CNVPutDataInBuffer () 函数将组包好的数据通过数据发布者发布到共享网络变量。

第五步，软件退出前，使用 CNVDispose () 函数释放创建数据发布者或发布缓存，最后使用 CNVFinish () 停止共享变量引擎。

2.2　上位机软件设计

上位机负责读取共享网络变量的数据并进行处理，是数据的订阅端。上位机软件的流程设计如图 6 所示。

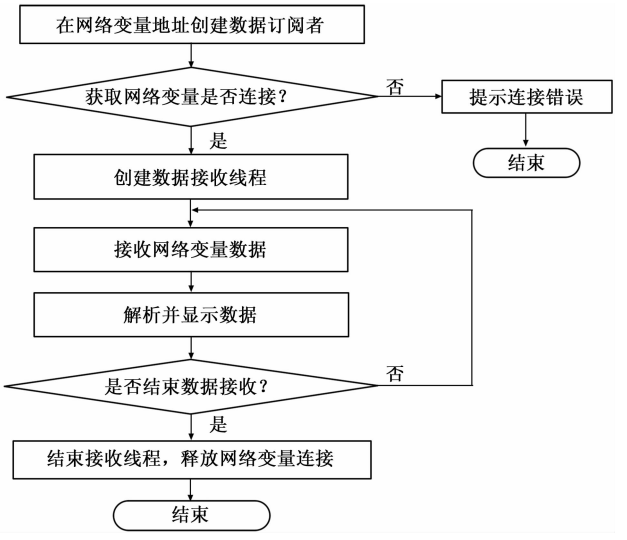

图 6　上位机软件设计流程图

第一步，使用 CNVCreateSubscriber () 函数在网络变量地址创建数据订阅者；如果数据量较大，则使用 CNVCreateBufferedSubscriber () 函数创建数据订阅缓存。通过使用缓冲区，可以解决对于一个变量读/写速度的波动问题。当读出操作偶尔比写入操作慢的情况可能会导致一些更新数据的丢失，如果应用可以容忍数据丢失，就不需要使用缓冲区；但是如果读取操作必须获得每个更新数据，就需要使用缓冲区[10]。

第二步，使用 CNVGetConnectionAttribute () 函数获得与网络变量的连接状态，如果连接异常，提示用户检查错误原因并退出；

第三步，创建数据接收线程，不断的从网络变量读取数据，解析数据并进行显示，同时将数据写入记录文件中存储。

从共享网络变量接收数据并提取出有效数据的方法与发布端相反，步骤如下：

1）使用 CNVGetDataFromBuffer () 函数检查网络变量是否有新数据到达；

2）使用 CNVGetNumberOfStructFields () 函数获取接收到的结构体数量，

3）使用 CNVGetStructFields () 函数获取结构体数据；

4）使用 CNVGetArrayDataValue () 函数获取结构体中的数组元素，使用 CNVGetScalarDataValue () 函数获取结构体中的数据值；

5）测试完成后使用 CNVDisposeData () 函数释放网络变量接收缓存；

如果不需要获得网络变量的每个更新数据，仅获取最新数据进行显示，那么上面的 5 个步骤可以简化为以下 3 个：

1）使用 CNVGetConnectionAttribute () 函数通过 CNVMostRecentDataAttribute 参数查询最新的结构体数据；

2）在定时器函数中重复调用 CNVGetArrayDataValue () 函数获取共享网络变量结构体中的数组元素，使用 CNVGetScalarDataValue () 函数获取数据值；

3）测试完成后使用 CNVDisposeData () 函数释放网络变量接收缓存。

最后，当用户结束数据接收操作时，软件停止数据接收线程，使用 CNVDispose () 函数释放创建数据订阅者或订阅缓存，最后使用 CNVFinish () 停止共享变量引擎。

2.3　上下位机软件交互协议

在下位机上完成共享网络变量数据的发布流程以及在上位机上完成网络变量数据的订阅流程后，还需要制定上下位机之间的通讯交互协议，由上位机向下位机发送控制命令，下位机根据控制命令开始或停止数据采集和发布。

上下位机软件之间的交互命令包括开始自检、停止自检、开始测试、停止测试和关闭软件等几个命令。这些命令采用 TCP 协议进行传输，这样做的好处是将交互命令和共享网络变量的数据传输相互分离，简化软件的开发和数据处理流程。

上下位机软件之间的命令交互流程如图 7 所示。

第一步，下位机启动后首先在指定端口创建 TCP Server 和共享网络变量，等待上位机进行连接；上位机启动后根据配置文件中定义的端口号连接 TCP Server，连接成功后在界面上显示已连接状态；

第二步，根据用户在上位机软件界面上的操作，通过 TCP 连接向下位机发送开始测试等控制命令，下位机收到命令后发送应答，开始数据采集并将数据更新到共享网络

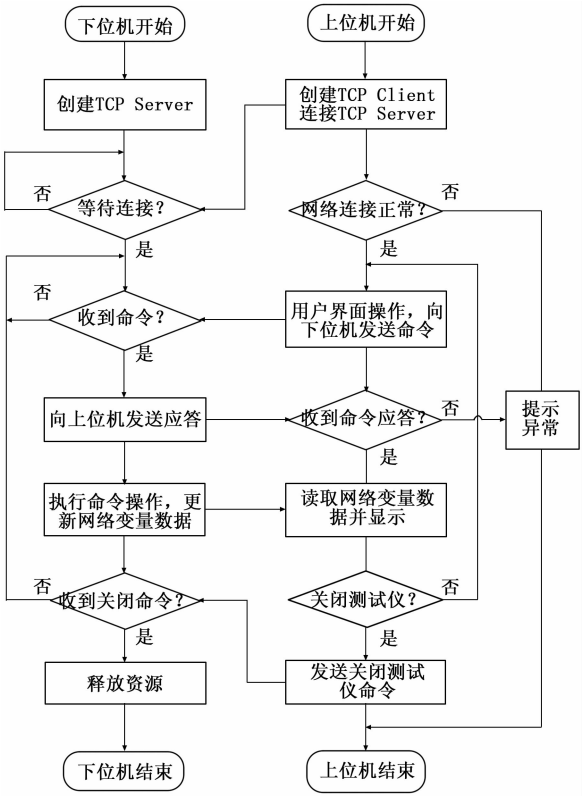

图 7 上下位机软件交互流程

变量中；上位机从共享网络变量中读取数据，完成解析并在软件界面中进行显示；

第三步，用户在上位机软件界面中点击停止测试按钮，软件通过 TCP 连接向下位机发送停止测试命令，下位机收到命令后发送应答，停止数据采集和共享变量的更新；

第四步，当用户点击退出按钮时，软件通过 TCP 连接向下位机发送关闭软件命令，下位机关闭网络共享变量和 TCP 连接，释放资源并退出。

3 功能验证

为了验证软件设计的可行性和 NI PSP 协议的传输效率，在测试系统中进行了上下位机的软件部署和软件功能的验证。

下位机没有图形化的软件界面，下位机软件运行后创建以时间命名的本地数据记录文件，将将采集到的数据保存在本地文件中。

上位机软件界面如图 8 所示，Tab 页面分别对应两个 RTU，这两个 RTU 通过各自的共享网络变量与上位机进行数据传输。用户点击“开始”按钮后，上位机软件向两个 RTU 的下位机软件发送开始测试命令，下位机软件将采集到的现场数据通过各自的共享网络变量发送给上位机；上位机分别从这两个共享变量读取数据，并对读取的数据进行解析，根据解析结果对 RTU 的每一个通道的状态进行显示，同时按照与下位机相同的格式将接收到的数据保存在数据记录文件中。

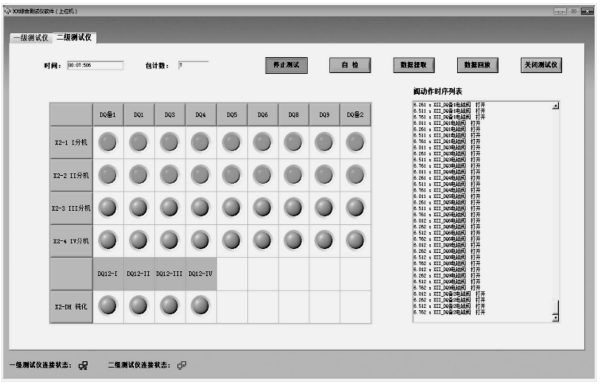

图 8 上位机软件界面

测试过程结束后，通过对比上、下位机中的保存的数据记录文件，验证了在测试过程中软件通过共享网络变量进行数据传输没有错误发生，验证了数据传输的可靠性。

4 结束语

本文介绍了以共享网络变量技术为基础，具有多节点大数据量传输能力的分布式测试软件的设计方法。与传统采用 Socket API 函数进行数据传输的设计方法相比，使用共享网络变量的编程更为简单，降低了软件开发的难度，可以使开发人员专注于测试功能的实现，提高软件开发的效率。

采用该设计方法开发的分布式测试软件已在某型号电磁阀测试系统中得到了应用，实际验证了共享网络变量的传输效率和可靠性，具有一定的推广价值。

参考文献：

[1] 张 荣，王 珏，周继昆，等. 总线类测试系统的技术现状及发展方向 [J]. 装备环境工程，2016，13 (5).

[2] 孙 露，王黎明. 基于 UDP/IP 协议分布式测试高速数据传输嵌入系统设计 [J]. 核电子学与探测技术，2014 (6).

[3] 任泰明. TCP/IP 协议与网络编程 [M]. 西安：西安电子科技大学出版社，2004.

[4] 许斯亮，吴小华，郑先成. 飞机供电系统的网络化分布式测试系统设计与实现 [J]. 测控技术，2006，25 (12)：72－74.

[5] Network Variable Technical Overview [EB/OL]. http：//www. ni. com/white－paper/5484/en/.

[6] 陈怀民，王龙福，吴成富，等. 基于 VxWorks 的分布式测试系统设计与实现 [J]. 测控技术，2009，28 (10)：71－74.

[7] Using the LabVIEW Shared Variable [EB/OL]. http：//www. ni. com/white－paper/4679/en/

[8] Adding Network Variables to a MeasurementStudio Windows Application [EB/OL]. http：//www. ni. com/tutorial/6593/en/

[9] NI－PSP Networking Technology [EB/OL]. LabWindows/CVI 2012 Help.

[10] Comparison of LabWindows/CVI Network Variable Connection Types [EB/OL]. www. ni. com/white－paper/7515/en.