[**虚拟仪器系统I/O接口软件——VISA** 1](#__RefHeading___Toc290220541)

[**VISA操作表** 4](#__RefHeading___Toc290220542)

[**读资源** 7](#__RefHeading___Toc290220543)

[**写资源** 38](#__RefHeading___Toc290220544)

[**VISA资源定义** 44](#__RefHeading___Toc290220545)

[**VISA应用实例** 73](#__RefHeading___Toc290220546)

[**续** 83](#__RefHeading___Toc290220547)

虚拟仪器系统I/O接口软件——VISA

|  |
| --- |
| VISA是虚拟仪器系统的I/O接口软件，为虚拟仪器系统的统一性和扩展性奠定了基础。本章介绍了VISA的概念和特点，论述了VISA的结构模型，重点分析了VISA资源与资源类的定义与描述，并结合实例，详细讨论了各资源的属性、事件与操作规范与软件实现。在本章结束，还讨论了VISA的实际设计思路与方法。  3.1 VISA的概念与特点  3.1.1 VISA的由来  随着虚拟仪器系统的出现与发展，I/O接口软件作为虚拟仪器系统软件结构中承上启下的一层，其模型化与标准化越来越重要。I/O接口软件驻留于虚拟仪器系统的系统管理器——计算机系统中，是实现计算机系统与仪器之间命令与数据传输的桥梁和纽带。许多仪器生产厂家在推出硬件接口电路的同时，也纷纷推出了不同结构的I/O接口软件，有的只针对某类仪器（如NI公司用于控制GPIB仪器的NI-488及用于控制VXI仪器的NI-VXI），有的在向统一化的方向靠拢（如HP公司的SICL—标准仪器控制语言），这些都是在仪器生产厂家内部通用的、优秀的I/O接口软件。  一般的I/O接口软件的结构都采用了自顶向下的设计模型：首先列出该I/O接口软件需要控制的所有仪器类型，然后列出了各类仪器的所有控制功能，最后将各类仪器控制功能中相同的操作功能尽可能地以统一的形式进行合并，并将统一的功能函数称为核心功能函数（如将GPIB仪器的读/写与RS232串行仪器的读/写统一为一个核心功能函数）。所有统一形式的核心函数与其它无法合并的、与仪器类型相关的操作功能函数一起构成了自顶向下的I/O接口软件，实现不同类型的仪器的互操作性与兼容性。然而，这种构成方法只适用于消息基器件的互操作性（如消息读、消息写、软件触发、状态获取、异步事件处理等功能），对于如中断处理、内存映射、接口配置、硬件触发等属于器件特有的操作，根本无法得到统一的核心函数，消息基器件与寄存器基器件无法在自顶向下的I/O接口软件中得到统一。核心函数集在整个I/O接口软件中只有一个小子集，特定操作函数集是一个大子集。自顶向下结构的I/O接口软件实质上是建立在仪器类型层的叠加，并没有真正实现接口软件的统一性。同时应该说，自顶向下的设计方法为真正统一的I/O接口软件的设计与实现提供了经验借鉴与尝试。VPP联盟在考察了多个I/O接口软件之后，提出了一种自底向上的I/O接口软件模型，也就是VISA。  3.1.2 VISA模型结构  VISA是虚拟仪器软件结构（Virtual Instrument Software Architecture）的缩写，实质是一个I/O接口软件及其规范的总称。一般情况下，将这个I/O接口软件称为VISA。  如上所说，VISA的构成是采用了自底向上的结构。与自顶向下的方法不同的是，VISA的实现首先定义了管理所有资源的资源（在这儿，资源的概念相当于面向对象程序设计中的对象，具体的定义与描述见下节），这个资源称为VISA资源管理器，它用于管理、控制与分配VISA资源的操作功能。各种操作功能主要包括：  1、 资源寻址；  2、 资源创建与删除；  3、 资源属性的读取与修改；  4、 操作激活；  5、 事件报告；  6、 并行与存取控制；  7、 缺省值设置。  第二步是在资源管理器基础上，列出了各种仪器各自的操作功能，并实现操作功能的合并。在这个基础上实现的资源实质可以包括不同格式的操作，如读资源包括了消息基器件的读，也包括了寄存器基器件的读；既可以包括同步读操作，又可以包括异步读操作。每一个资源内部，实质是各种操作的集合。这种资源在VISA中即为仪器控制资源，包含各种仪器操作的资源称为通用资源，而将无法合并的功能，称为特定仪器资源。  第三步，需要定义与创建一个用API实现的资源，为用户提供单一的控制所有VISA仪器控制资源的方法，在VISA中称为仪器控制资源组织器。  与自顶向下的构成方式相比，VISA的构成模型是从仪器操作本身开始的，它实现的统一是深入到操作功能中去而不是停留于仪器类型之上。在VISA的结构中，仪器类型的区别体现到统一格式的资源中的操作的选取，对于VISA使用者来说，形式上与用法上是单一的。在理论层次上，自顶向下的方法属于归纳范畴，而自底向上的方法则属于演绎范畴。因此，自顶向下是对过去所有仪器类型的总结，而不可能提供扩展接口，而自底向上的结构是从共性到个性的推广，它的兼容性不仅仅是过去、现在，还可以包括将来。正由于这种自底向上的设计方法，VISA为虚拟仪器系统软件结构提供了一个共同的、统一的基础，来自于不同供应厂家的不同的仪器软件，可以运行于同一平台之上了。  VISA的结构模型如图3.1所示。  图3.1 VISA结构模型概图  VISA结构模型自下往上，构成一个金字塔结构，最底层为资源管理器，其上为I/O级资源、仪器级资源与用户自定义资源集。其中，用户自定义资源集的定义，在VISA规范中并没有规定，它是VISA的可变层，实现了VISA的可扩展性与灵活性，而在金字塔顶的用户层应用，是用户利用VISA资源实现的应用程序，其本身并不属于VISA资源。  3.1.3 VISA的特点  基于自底向上结构模型的VISA创造了一个统一形式的I/O控制函数集，它是所有现存的I/O接口软件的功能超集，在形式上与其它I/O接口软件十分相似。一方面，对于初学者或是简单任务的设计者来说，VISA提供了简单易用的控制函数集，在应用形式上相当简单。另一方面，对于复杂系统的组建者来说，VISA提供了非常强大的仪器控制功能与资源管理功能，能提供仪器间的互操作性与兼容性。它是易用性、可扩展性、互操作性、兼容性的完全统一体。  与其它现存的I/O接口软件相比，VISA具有以下几个特点：  1、 VISA的I/O控制功能适用于各种仪器类型，VISA包含了VXI仪器、GPIB仪器、RS232串行接口仪器等各类仪器的控制操作，也包含了消息基器件、寄存器基器件、存贮器件等仪器的操作，其形式上是统一的。  2、 VISA的I/O控制功能适用于各种仪器硬件接口类型，以VXI仪器系统为例，无论采用嵌入式计算机结构、GPIB接口外挂式结构、MXI接口外挂式结构，还是采用IEEE1394接口外挂式结构，对于VXI仪器的操作函数是一样的。同样，无论VXI仪器在系统中的逻辑地址是多少，仪器操作函数也是一样的。  3、 VISA的I/O控制功能适用于单处理器系统结构，也适用于多处理器结构或分布式网络结构。  4、 VISA的I/O控制功能适用于多种网络机制，无论虚拟仪器系统网络构成为VXI多机箱扩展网络还是以太网，仪器操作是一致的。  5、 VISA的I/O软件库的源代码是唯一的，其与操作系统及编程语言无关，只是提供了不同形式的API文件作为系统的引出。  由于VISA结构考虑到了多种仪器类型与网络机制的兼容性，因此以VISA I/O接口软件为基础的虚拟仪器系统，不仅可以与过去已有的仪器系统（如GPIB仪器系统及串行接口仪器系统）结合，也完全可以将仪器系统从过去的集中式结构过渡到分布式结构。VISA的兼容性与互操作性，保证了新一代仪器完全可以加入到虚拟仪器系统中去，同时也保证了仪器系统的投资者不会因为新仪器的出现而将过去的系统抛弃，从而可以不使投资浪费。系统集成时，不必再选择某家特殊的软件和硬件产品，可以根据自己的需要，在所有的VPP产品中作出最佳选择，系统的兼容性与强健性大大增强了，系统的标准化与统一性也找到了最基础的保障。 |

VISA操作表

|  |
| --- |
| 操作表：  1、VISA资源模板：  viClose（vi）：关闭特定的对话通道。  viGetAttribute（vi，attribute，attrState）：获取资源属性状态值。  viSetAttribute（vi，attribute，attrState）：设置资源属性状态值。  viStatusDesc（vi，status，desc）：获取返回状态描述字符串。  viTerminate（vi，degree，jobId）：请求VISA资源终止一个或所有对话通道的正常运行。  viLock（vi，lockType，timeout，requestId，accessKey）：设置资源存取模式。  viUnlock（vi）：取消资源存取模式。  viEnableEvent（vi，eventType，mechanism，context）：允许特定事件通知。  viDisableEvent（vi，eventType，mechanism）：不允许特定事件通知。  viDiscardEvents（vi，eventType，mechanism）：刷新一个对话通道上事件发生。  viWaitOnEvent（vi，ineventTypeList，timeout，outEventType，outContext）：等待特定事件的发生。  viInstallHandler（vi，eventType，handler，userHandle）：安装回调事件句柄 。  viUnInstallHandler（vi，eventType，handler，userHandle）：卸载回调事件句柄。  2、VISA资源管理器：  viOpenDefaultRM（sesn）：打开缺省资源管理器资源对话通道。  viOpen（sesn，rsrcname，accessMode，timeout，vi）：打开特定资源的对话通道。  viFindRsrc（sesn，expr，findList，retcnt，instrDesc）：查询VISA系统进行资源定位。  viFindNext（findList，instrDesc）：返回前一个查询操作查得的资源。  3、仪器控制管理：  viRead（vi，buf，count，retCount）：从器件同步读取数据。  viReadAsync（vi，buf，count，jobId）：从器件异步读取数据。  viWrite（vi，buf，count，retCount）：将数据同步写入到器件中。  viWriteAsync（vi，buf，count，jobId）：将数据异步写入到器件中。  viAssertTrigger（vi，protocol）：用特定协议确认硬件或软件触发。  viReadSTB（vi，status）：读取服务请求状态字节。  viClear（vi）：清除器件。  viSetBuf（vi，mask，size）：设置格式化I/O缓冲区大小。  viFlush（vi，mask）：手动刷新格式化I/O缓冲区。  viPrintf（vi，writeFmt，arg1，arg2…）：按设定格式将数据传送到器件中。  viVPrintf（vi，writeFmt，params）：按设定格式将数据传送到器件中。  viScanf（vi，readFmt，arg1，arg2…）：按设定格式从器件中读取数据。  viVScanf（vi，readFmt，params）：按设定格式从器件中读取数据。  viQuery（vi，writeFmt，readFmt，arg1，arg2…）：按设定格式对器件进行数据读写。  viVQuery（vi，writeFmt，readFmt，params）：按设定格式对器件进行数据读写。  viIn8（vi，space，offset，value）：从接口总线读取8位（字节）单位。  viIn16（vi，space，offset，value）：从接口总线读取16位（字）单位数据。  viIn32（vi，space，offset，value）：从接口总线读取32位（双字）单位数据。  viOut8（vi，space，offset，value）：向接口总线写入8位（字节）单位数据。  viOut16（vi，space，offset，value）：向接口总线写入16位（字）单位数据。  viOut32（vi，space，offset，value）：向接口总线写入32位（双字）单位数据。  viMoveIn8（vi，space，offset，length，buf8）：从器件存储器向当地存储器移动8位（字节）单位数据。  viMoveIn16（vi，space，offset，length，buf8）：从器件存储器向当地存储器移动16位（字）单位数据。  viMoveIn32（vi，space，offset，length，buf8）：从器件存储器向当地存储器移动32位（双字）单位数据。  viMoveOut8（vi，space，offset，length，buf8）：从当地存储器向器件存储器移动8位（字节）单位数据。  viMoveOut16（vi，space，offset，length，buf8）：从当地存储器向器件存储器移动16位（字）单位数据。  viMoveOut32（vi，space，offset，length，buf8）：从当地存储器向器件存储器移动32位（双字）单位数据。  viMapAddress（vi，mapSpace，mapBase，mapSize，access，suggested，address）：映射内存空间。  viUnMapAddress（vi）：取消内存映射。  viPeek8（vi，addr，val8）：从特定地址读8位数据。  viPeek16（vi，addr，val16）：从特定地址读16位数据。  viPeek32（vi，addr，val32）：从特定地址读32位数据。  viPoke8（vi，addr，val8）：向特定地址写8位数据。  viPoke16（vi，addr，val16）：向特定地址写16位数据。  viPoke32（vi，addr，val32）：向特定地址写32位数据。  viMemAlloc（vi，size，offset）：从器件存储器分配内存。  viMemFree（vi，offset）：释放内存分配。 |

读资源

|  |
| --- |
| n   读资源（VI\_RSRC\_RD）：  1. 资源概述：控制器从器件读出任意数据块，控制器可以将接收到的数据块解释为消息、命令或二进制编码数据。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_FDC\_ACCESS\_MODE FDC(高速数据通道)缓冲区存取模式  VI\_ATTR\_FDC1\_AVAIL FDC版本1是否有效  VI\_ATTR\_FDC2\_AVAIL FDC版本2是否有效  VI\_ATTR\_FDC\_CHNL    FDC数据传送通道  VI\_ATTR\_FDC\_GEN\_SIGNAL\_EN   是否允许通过FDC传送数据  VI\_ATTR\_FDC\_MODE    FDC模式  VI\_ATTR\_FDC\_USE\_PAIR    一对或一个FDC有效  VI\_ATTR\_FDC\_USE\_VER FDC协议版本  VI\_ATTR\_PHYS\_ADDR   器件地址  VI\_ATTR\_7\_8\_BIT\_CMP 终止符有效位  VI\_ATTR\_NRDY\_ABORT\_EN   器件不再输出数据时是否被通知  VI\_ATTR\_OUTP\_RDY\_NOTIFY\_EN 数据有效是否被通知  VI\_ATTR\_RD\_PROT 传送协议  VI\_ATTR\_REPEAT\_ADDR\_EN 是否使用重复地址  VI\_ATTR\_SUPPRESS\_END\_ON 终止符是否被禁止  VI\_ATTR\_TERMCHAR    终止符  VI\_ATTR\_TERMCHAR\_EN 终止符是否被允许  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  VI\_ATTR\_TRANSFER\_MECH   传送机制(DMA或PIO)  VI\_ATTR\_CMDR\_SERV   主从模式  VI\_ATTR\_ASRL\_BAUD   波特率  VI\_ATTR\_ASRL\_DATA\_BITS 数据位  VI\_ATTR\_ASRL\_PARITY 校验极性  VI\_ATTR\_ASRL\_STOP\_BITS 停止位  VI\_ATTR\_ASRL\_FLOW\_CNTRL 数据流控制状态  3. 读资源定义的事件如下所示：  事件    说明  VI\_EVENT\_IO\_COMPLETION 异步I/O操作已完成  4. 读资源定义的操作如下：  viRead（vi，buf，count，retCount）  viReadAsync（vi，buf，count，jobId）  viReadStatus（vi，jobId，jobStatus）  4.1.    viRead（vi，buf，count，retCount）  1） 目标：从器件同步读取数据  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  buf 输出    ViBuf   数据块位置  count   输入    ViUInt32    读出的字节数  retCount    输出    ViPUInt32   实际传送字节数  3） 返回状态值：  完成代码：  VI\_SUCCESS 同步读数据完成  VI\_SUCCESS\_TERM\_CHAR    特定终止符被读取  VI\_SUCCESS\_MAX\_CNT 读取的字节数与count数相同  VI\_WARN\_NRDY\_ABORT 传送退出，器件未准备好输出数据  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_RAW\_WR\_PROT    传送时写协议被破坏  \_VIOL  VI\_ERROR\_RAW\_RD\_PROT    传送时读协议被破坏  \_VIOL  VI\_ERROR\_OUTP\_PROT\_VIOL 传送时输出协议错误  VI\_ERROR\_INP\_PROT\_VIOL 传送时输入协议错误  VI\_ERROR\_BERR   传送时总线错误  VI\_ERROR\_ABORT 传送时用户退出  VI\_ERROR\_IN\_PROGRESS    同步写已执行  VI\_ERROR\_INV\_SETUP 设置错误，不能开始写操作  4） 描述：该操作将从器件中读取的值放在buf，任一读资源通道只能登录一个同步读操作。  5） 相关项：参见viWriteAsync（）、viTerminate（）、viWriteStatus（）、viReadAsync（）、viReadStatus（）。  6） 实现要求：当满足以下三种情况的一种或几种时，被认为读操作完成：一是结束标号被读取，二是特定终止符被读取，三是读取的字节数与所需字节数相同。  4.2.    viReadAsync（vi，buf，count，jobId）  1） 目标：从器件异步读取数据  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  buf 输出    ViBuf   数据块位置  count   输入    ViUInt32    读出的字节数  jobId   输出    ViPjobId    异步读操作作业标识符  3） 返回状态值：  完成代码：  VI\_SUCCESS 异步读操作完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_QUEUE\_ERROR    不能进行读操作排队  4） 描述：该操作将从器件读取的数据放于buf中，任一读资源通道可以登录多个异步读操作，进行排队处理。  5） 相关项：参见viWrite（）、viTerminate（）、viWriteAsync（）、viRead（）、viReadStatus（）。  6） 实现要求：无。  4.3.    viReadStatus（vi，jobId，jobStatus）  1） 目标：获取读操作状态  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  jobId   输入    VijobId 读操作作业标识符  jobStatus   输出    ViPJobstatus    读操作状态  3） 返回状态值：  完成代码：  VI\_SUCCESS 状态获取完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_JOBID 作业标识符不当  4） 描述：当操作为同步时，标识符唯一确定，为VI\_SYNC；当操作为异步时，标识符为操作的输出参数值。  5） 相关项：参见viReadAsync（）。  6） 实现要求：无。  n   格式化I/O资源（VI\_RSRC\_FMT\_IO）：  1. 资源概述：数据流格式化I/O操作，可进行格式化的器件读写操作，并可进行缓冲区的I/O格式化。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_7\_8\_BIT\_CMP 终止符位数  VI\_ATTR\_END\_ON\_TERMCHAR\_EN 在GPIB系统中，EOI终止符是否被确认  VI\_ATTR\_NRDY\_ABORT 器件不再输出数据时是否被通知  VI\_ATTR\_PROT    读/写协议  VI\_ATTR\_RD\_BUF\_OPER\_MODE    读缓冲区操作模式  VI\_ATTR\_RD\_SIZE 读缓冲区大小  VI\_ATTR\_REPEAT\_ADDR 是否使用重复地址  VI\_ATTR\_RETURN\_MODE 读/写操作返回模式  VI\_ATTR\_SEND\_END\_EN 是否将END作为终止符  VI\_ATTR\_SUPPRESS\_END    是否禁止END作为终止符  VI\_ATTR\_TERMCHAR    终止符  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  VI\_ATTR\_TRANSFER\_MECH   传送机制(DMA或PIO)  VI\_ATTR\_WR\_BUF\_OPER\_MODE    写缓冲区操作模式  VI\_ATTR\_WR\_BUF\_SIZE 写缓冲区大小  VI\_ATTR\_ASRL\_BAUD   波特率  VI\_ATTR\_ASRL\_DATA\_BITS 数据位  VI\_ATTR\_ASRL\_PARITY 校验极性  VI\_ATTR\_ASRL\_STOP\_BITS 停止位  VI\_ATTR\_ASRL\_FLOW\_CNTRL 数据流控制状态  3. 格式化I/O资源定义的事件如下所示：无  4. 格式化I/O资源定义的操作如下：  viSetBuf（vi，mask，size）  viSetUBuf（vi，mask，size，buf）  viFlush（vi，mask）  viPrintf（vi，writeFmt，arg1，arg2…）  viVPrintf（vi，writeFmt，params）  viSPrintf（vi，buf，writeFmt，arg1，arg2…）  viSVPrintf（vi，buf，writeFmt，params）  viBWrite（vi，buf，count，retCount）  viScanf（vi，readFmt，arg1，arg2…）  viVScanf（vi，readFmt，params）  viSScanf（vi，buf，readFmt，arg1，arg2…）  viSVScanf（vi，buf，readFmt，params）  viBRead（vi，buf，count，retCount）  viQueryf（vi，writeFmt，readFmt，arg1，arg2…）  viVQueryf（vi，writeFmt，readFmt，params）  4.1.    viSetBuf（vi，mask，size）  1） 目标：设置格式化I/O缓冲区大小  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  mask    输入    ViPUInt16   缓冲区类型  size    输入    ViPUInt32   缓冲区大小  3） 返回状态值：  完成代码：  VI\_SUCCESS 缓冲区大小设置完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_ALLOC\_MEM 系统无法分配缓冲区  VI\_ERROR\_INV\_MASK   缓冲区类型设置错误  4） 描述：该操作改变读/写缓冲区大小，缓冲区类型可取VI\_READ及VI\_WRITE。  5） 相关项：参见viSefUBuf（）、viFlush（）。  6） 实现要求：调用该操作会刷新相关缓冲区。  4.2.    viSetUBuf（vi，mask，size，buf）  1） 目标：设置格式化I/O用户缓冲区  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  mask    输入    ViPUInt16   缓冲区类型  size    输入    ViPUInt32   缓冲区大小  buf 输入    ViBuf   实际缓冲区地址  3） 返回状态值：  完成代码：  VI\_SUCCESS 用户缓冲区设置完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_MASK   缓冲区类型设置错误  4） 描述：该操作为格式化I/O开辟一个缓冲区，该缓冲区不能用作其它用途。  5） 相关项：参见viSefBuf（）。  6） 实现要求：调用该操作会刷新相关缓冲区。  4.3.    viFlush（vi，mask）  1） 目标：手工刷新格式化I/O缓冲区  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  mask    输入    ViPUInt16   缓冲区类型  3） 返回状态值：  完成代码：  VI\_SUCCESS 缓冲区刷新完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_IO I/O错误  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_ABORT 在操作结束前收到退出信号  VI\_ERROR\_INV\_MASK   缓冲区类型设置错误  4） 描述：该操作手动刷新相关读/写缓冲区内容。  5） 相关项：参见viSefBuf（）、viSefUBuf（）。  6） 实现要求：若对一空缓冲区进行刷新操作，相当于没有动作。  4.4.    viPrintf（vi，writeFmt，arg1，arg2…）  1） 目标：按设定格式将数据传送到器件中  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  writeFmt    输入    ViString    参数格式  arg1，arg2…    输入        各个所需参数  3） 返回状态值：  完成代码：  VI\_SUCCESS 参数成功地格式化  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_IO I/O错误  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_ABORT 在操作结束前收到退出信号  VI\_ERROR\_FILE\_NOPEN 文件没有打开或文件存取特性不当  VI\_ERROR\_NSUP\_FMT   参数格式不当  4） 描述：该操作以一定格式将数据传送到器件中，viWrite（）实际上是完成低层I/O操作。任一对话通道不能同时进行viWrite（）及viPrintf（）操作。  5） 相关项：参见viSPrintf（）、viVPrintf（）、viSVPrintf（）、viBWrite（）。  6） 实现要求：输入参数与其格式必须匹配。  4.5.    viSPrintf（vi，buf，writeFmt，arg1，arg2…）：与viPrintf（）不同的只在于viSPrintf（）将数据写入buf参数的缓冲区而非器件中。  4.6.    viVPrintf（vi，writeFmt，params）：与viPrintf（）不同的只在于viVPrintf（）中参数以列表形式给出。  4.7.    viSVPrintf（vi，buf，writeFmt，params）：与viPrintf（）不同的只在于viSVPrintf（）中参数以列表形式给出，并写入到缓冲区中。  4.8.    viBWrite（vi，buf，count，retCount）：与viWrite（）不同的只在于viBWrite（）将数据写入到缓冲区而不是器件中去。  4.9.    viScanf（vi，readFmt，arg1，arg2…）  1） 目标：按设定格式从器件中读取数据  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  readFmt 输入    ViString    参数格式  arg1，arg2…    输出        各个所得参数值  3） 返回状态值：  完成代码：  VI\_SUCCESS 数据成功地格式化并被读取  VI\_SUCCESS\_END\_RECV END标志被接收到  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_IO I/O错误  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_ABORT 在操作结束前收到退出信号  VI\_ERROR\_FILE\_NOPEN 文件没有打开或文件存取特性不当  VI\_ERROR\_NSUP\_FMT   参数格式不当  4） 描述：该操作以一定格式从器件中读取数据，viRead（）实际上是完成低层I/O操作。任一对话通道不能同时进行viRead（）及viScanf（）操作。  5） 相关项：参见viSScanf（）、viVScanf（）、viSVScanf（）、viBRead（）。  6） 实现要求：输出参数与其格式必须匹配。  4.10.   viSScanf（vi，buf，readFmt，arg1，arg2…）：与viScanf（）不同的只在于viSScanf（）从buf参数的缓冲区中读出数据而非从器件中读取。  4.11.   viVScanf（vi，readFmt，params）：与viScanf（）不同的只在于viVScanf（）中参数以列表形式给出。  4.12.   viSVScanf（vi，buf，readFmt，params）：与viScanf（）不同的只在于viSVScanf（）中参数以列表形式给出，并从缓冲区中读取。  4.13.   viBRead（vi，buf，count，retCount）：与viRead（）不同的只在于viBRead（）从缓冲区而不是从器件中读取数据。  4.14.   viQuery（vi，writeFmt，readFmt，arg1，arg2…）  1） 目标：按设定格式对器件进行数据读写  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  writeFmt    输入    ViString    写参数格式  readFmt 输入    ViString    读参数格式  arg1，arg2…    输入、输出      各个所需或所得参数值  3） 返回状态值：  完成代码：  VI\_SUCCESS 参数成功地格式化并被读取或写入  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_IO I/O错误  VI\_ERROR\_INV\_FMT    格式错误  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_ABORT 在操作结束前收到退出信号  VI\_ERROR\_FILE\_NOPEN 文件没有打开或文件存取特性不当  VI\_ERROR\_NSUP\_FMT   参数格式不合法  4） 描述：该操作与器件进行数据交换，输入或输出是需要根据参数格式类型来定的，实际上包括了读与写操作。  5） 相关项：参见viBRead（）、viBWrite（）、viVQueryf（）。  6） 实现要求：输入/输出参数与其格式必须匹配。  4.15.   viVQueryf（vi，writeFmt，readFmt，params）：与viQueryf（）不同的是viVQueryf（）参数以列表形式给出。  n   触发资源（VI\_RSRC\_TRIG）：  1. 资源概述：监视与控制器件及接口的触发进程。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_HW\_TRIG\_SENSE\_EN    触发是否会引发CPU中断  VI\_ATTR\_TRIG\_ASSERT\_MODE    触发确认模式  VI\_ATTR\_TRIG\_MAP\_ROUTE 触发映射  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  VI\_ATTR\_TRIG\_LINE   当前资源特定触发线  3. 触发资源定义的事件如下所示：  事件    说明  VI\_EVENT\_TRIG   硬件触发产生  4. 触发资源定义的操作如下：  viAssertTrigger（vi，protocol）  viMapTrigger（vi，srcTrig，destTrig，mode）  viUnmapTrigger（vi，srcTrig，destTrig）  4.1.    viAssertTrigger（vi，protocol）  1） 目标：用特定协议确认硬件或软件触发  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  protocol    输入    ViUInt16    触发协议  3） 返回状态值：  完成代码：  VI\_SUCCESS 特定触发确认  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_PROT   协议不当  VI\_ERROR\_NSUP\_OPER 硬件不支持  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_LINE\_IN\_USE    触发线正被使用  VI\_ERROR\_LINE\_NCONFIG   触发线未配置  VI\_ERROR\_OPER\_NCOMPLETE 上一个操作未完成  VI\_ERROR\_ACK\_PENDING    上个响应正在登录，使这个操作退出  4） 描述：该操作包括的触发协议有：缺省协议、协议开启、协议关闭、协议开始、协议结束、协议同步、协议半同步、协议异步、协议半同步并等待响应、协议异步并等待响应等。  5） 相关项：参见viMapTrigger（）、viUnmapTrigger（）。  6） 实现要求：无。  4.2.    viMapTrigger（vi，trig，direction，mode）  1） 目标：将特定触发、时钟或前面板信号映射到另一信号  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  trig    输入    ViString    映射源或目标触发线  direction   输入    ViUInt16    映射方向  mode    输入    ViUInt16    信号调理方式  3） 返回状态值：  完成代码：  VI\_SUCCESS 两个信号映射成功  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_NSUP\_MODE 不支持特定模式  VI\_ERROR\_NUSP\_SRC\_TRIG 触发信号源不支持  VI\_ERROR\_NUSP\_DEST\_TRIG 触发信号目标不支持  VI\_ERROR\_TRIG\_MAPPED    目标触发器已被映射  4） 描述：映射信号包括特定触发信号、时钟信号或外部信号等。  5） 相关项：参见viUnmapTrigger（）、viAssertTrigger（）。  6） 实现要求：无。  4.3.    viUnmapTrigger（vi，trig）  1） 目标：取消信号之间的映射  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  trig    输入    ViString    被映射的触发线  3） 返回状态值：  完成代码：  VI\_SUCCESS 两个信号映射取消成功  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_NUSP\_SRC\_TRIG 触发信号源不支持  VI\_ERROR\_NUSP\_DEST\_TRIG 触发信号目标不支持  VI\_ERROR\_TRIG\_NMAPPED   特定触发信号当前未被映射  4） 描述：映射信号包括特定触发信号、时钟信号或外部信号等。  5） 相关项：参见viMapTrigger（）、viAssertTrigger（）。  6） 实现要求：无。  n   状态/服务请求资源（VI\_RSRC\_SRQ）：  1. 资源概述：包括两个作用：一是作为服务请求的响应者，可监测系统的服务请求并获得状态信息；二是作为服务请求的发生者，可产生服务请求并提供状态信息。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_AUTOPOLL\_EN 资源操作模式  VI\_ATTR\_STB 服务请求状态  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  3. 状态/服务请求资源定义的事件如下所示：  事件    说明  VI\_EVENT\_SERVICE\_REQ    服务请求通知事件  4. 状态/服务请求资源定义的操作如下：  viReadSTB（vi，status）  viRequestServ（vi，request，status）  4.1.    viReadSTB（vi，status）  1） 目标：读取服务请求状态字节  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  status 输出    ViUInt16    服务请求状态  3） 返回状态值：  完成代码：  VI\_SUCCESS 服务请求状态字节读取完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_SRQ\_NOCCURRED 服务请求未收到  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_BERR   总线错误  4） 描述：该操作从一个服务请求者（消息基器件）中读取服务请求状态。  5） 相关项：参见viEventHandler（）、viSetAttribute（）、viGetAttribute（）。  6） 实现要求：无。  4.2.    viRequestServ（vi，request，status）  1） 目标：向服务提供者（命令者）确认服务请求  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  request 输入    ViBoolean   服务请求目的  status 输入    ViUInt16    服务请求状态  3） 返回状态值：  完成代码：  VI\_SUCCESS 服务请求完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_SERVICE\_REQ    服务请求状态字节与请求参数不一致  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_BERR   总线错误  4） 描述：该操作用于当前器件向服务提供者发送服务请求。  5） 相关项：参见VI\_EVENT\_SRQ\_SERVICED事件。  6） 实现要求：无。  n   清除资源（VI\_RSRC\_CLR）：  1. 资源概述：向器件发送清除命令。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_HW\_SENSE\_EN 硬件是否对接收到指令敏感  VI\_ATTR\_PHYS\_ADDR   器件地址  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  3. 清除资源定义的事件如下所示：  事件    说明  VI\_EVENT\_CLEAR 器件清除命令接收事件  4. 清除资源定义的操作如下：  viClear（vi）  4.1.    viClear（vi）  1） 目标：清除器件  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  3） 返回状态值：  完成代码：  VI\_SUCCESS 器件清除完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_BERR   总线错误  VI\_ERROR\_TMO    超时错误  4） 描述：该操作向器件发送一个清除命令。  5） 相关项：无。  6） 实现要求：无。  n   高级存取资源（VI\_RSRC\_HL\_ACC）：  1. 资源概述：在接口级对器件的寄存器进行存取操作。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_BUS\_TYPE    接口总线类型  VI\_ATTR\_PHYS\_ADDR   器件地址  VI\_ATTR\_TYPE    高级存取对象类型（器件还是接口）  VI\_ATTR\_DEST\_ACCESS\_PRIV    写操作特性  VI\_ATTR\_DEST\_BYTE\_ORDER 写操作字节顺序  VI\_ATTR\_DO\_RETRY    是否允许软件重入  VI\_ATTR\_SRC\_ACCESS\_PRIV 读操作特性  VI\_ATTR\_SRC\_BYTE\_ORDER 读操作字节顺序  VI\_ATTR\_USE\_BLOCK   是否允许块传送  VI\_ATTR\_TRANSFER\_MECH   传送机制  VI\_ATTR\_SRC\_INCREMENT   源偏移量  VI\_ATTR\_DEST\_INCREMENT 目标偏移量  3. 高级存取资源定义的事件如下所示：无  4. 高级存取资源定义的操作如下：  viIn（vi，space，offset，width，value）  viIn8（vi，space，offset，value）  viIn16（vi，space，offset，value）  viIn32（vi，space，offset，value）  viIn64（vi，space，offset，value）  viOut（vi，space，offset，width，value）  viOut8（vi，space，offset，value）  viOut16（vi，space，offset，value）  viOut32（vi，space，offset，value）  viOut64（vi，space，offset，value）  viMove（vi，srcspace，srcOffset，srcWidth，destspace，destOffset，  destWidth，length）  4.1.    viIn（vi，space，offset，width，value）  viIn8（vi，space，offset，value）  viIn16（vi，space，offset，value）  viIn32（vi，space，offset，value）  viIn64（vi，space，offset，value）  1） 目标：从接口总线读取8，16，32或64位字  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  space   输入    ViUInt16    地址空间类型  offset 输入    ViUInt32    读取接口/器件偏移量  width   输入    ViUInt16    读取数据长度  value   输出    ViBuf、ViPUInt(8、16、32、64)   读取数据  3） 返回状态值：  完成代码：  VI\_SUCCESS 读操作完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_BERR   总线错误  VI\_ERROR\_INV\_SPACE 地址空间不合法  VI\_ERROR\_INV\_OFFSET 偏移量不合法  VI\_ERROR\_NSUP\_BYTE\_ORDER    不支持特定字节顺序  VI\_ERROR\_NSUP\_OFFSET    硬件不支持特定地址偏移量  VI\_ERROR\_NSUP\_PRIV 硬件不支持存取特性  4） 描述：该操作用特定数据长度与地址空间读取位于一定地址的数据值。其中viIn8（）、viIn16（）、viIn32（）、viIn64分别规定数据长度为8、16、32与64位。  5） 相关项：viOut（）。  6） 实现要求：参数value值viBuf对应viIn（），viPUInt8对应viIn8（），viPUInt16对应viIn16（），viPUInt32对应viIn32（），viPUInt64对应viIn64（）。  4.2.    viOut（vi，space，offset，width，value）  viOut8（vi，space，offset，value）  viOut16（vi，space，offset，value）  viOut32（vi，space，offset，value）  viOut64（vi，space，offset，value）  1） 目标：向接口总线写入8，16，32或64位字  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  space   输入    ViUInt16    地址空间类型  offset 输入    ViUInt32    写入接口/器件偏移量  width   输入    ViUInt16    写入数据长度  value   输入    ViBuf、ViPUInt(8、16、32、64)   写入数据  3） 返回状态值：  完成代码：  VI\_SUCCESS 写操作完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_BERR   总线错误  VI\_ERROR\_INV\_SPACE 地址空间不当  VI\_ERROR\_INV\_OFFSET 偏移量不当  VI\_ERROR\_NSUP\_BYTE\_ORDER    不支持特定字节顺序  VI\_ERROR\_NSUP\_OFFSET    硬件不支持特定地址偏移量  VI\_ERROR\_NSUP\_PRIV 硬件不支持存取特性  4） 描述：该操作用特定数据长度与地址空间将数据写入到一定地址中。其中viOut8（）、viOut16（）、viOut32（）、viOut64分别规定数据长度为8、16、32与64位。  5） 相关项：viIn（）。  6） 实现要求：参数value值viBuf对应viOut（），viPUInt8对应viOut8（），viPUInt16对应viOut16（），viPUInt32对应viOut32（），viPUInt64对应viOut64（）。  4.3.    viMove（vi，srcSpace，srcOffset，srcWidth，destSpace，destOffset，destWidth，length）  1） 目标：移动数据块  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  srcSpace    输入    ViUInt16    源空间地址  srcOffset   输入    ViUInt32    源偏移量  srcWidth    输入    ViUInt16    源数据长度  destSpace   输入    ViUInt16    目标空间地址  destOffset 输入    ViUInt32    目标偏移量  destWidth   输入    ViUInt16    目标数据长度  length 输入    ViUInt32    传送数据个数  3） 返回状态值：  完成代码：  VI\_SUCCESS 数据块移动完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_BERR   总线错误  VI\_ERROR\_INV\_SPACE 地址空间不当  VI\_ERROR\_INV\_OFFSET 偏移量不当  VI\_ERROR\_INV\_WIDTH 数据长度不当  VI\_ERROR\_NSUP\_BYTE\_ORDER    不支持特定字节顺序  VI\_ERROR\_NSUP\_OFFSET    硬件不支持特定地址偏移量  VI\_ERROR\_NSUP\_VAR\_WIDTH 源与目标数据长度不一致  VI\_ERROR\_NSUP\_PRIV 硬件不支持存取特性  4） 描述：该操作将数据从源地址移动到目标地址。源与目标数据长度必须一致。  5） 相关项：无。  6） 实现要求：无。  n   低级存取资源（VI\_RSRC\_LL\_ACC）：  1. 资源概述：在寄存器级对器件的寄存器进行存取操作。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_BUS\_TYPE    接口总线类型  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  VI\_ATTR\_ACC\_PRIV    存取特性  VI\_ATTR\_ADDR\_SPACE 地址空间  VI\_ATTR\_BLOCK   块传送允许  VI\_ATTR\_BYTE\_PREFETCH   是否允许预取操作  VI\_ATTR\_VIRT\_BASE   虚拟基地址  VI\_ATTR\_WIN\_BASE\_ADDR   总线基地址  VI\_ATTR\_WIN\_OWNER   用户存取特性  VI\_ATTR\_WR\_POST 写登记是否允许  VI\_ATTR\_A16\_SPC\_INFO    A16空间映射信息  VI\_ATTR\_A24\_SPC\_INFO    A24空间映射信息  VI\_ATTR\_A32\_SPC\_INFO    A32空间映射信息  3. 低级存取资源定义的事件如下所示：  事件    说明  VI\_EVENT\_BERR   总线错误发生事件  4. 低级存取资源定义的操作如下：  viMap（vi，mapSpace，mapBase，mapSize，owner，suggested，address）  viUnMap（vi）  viPoke（vi，addr，width，value）  viPoke8（vi，addr，value）  viPoke16（vi，addr，value）  viPoke32（vi，addr，value）  viPoke64（vi，addr，value）  viPeek（vi，addr，width，value）  viPeek8（vi，addr，value）  viPeek16（vi，addr，value）  viPeek32（vi，addr，value）  viPeek64（vi，addr，value）  4.1.    viMap（vi，mapSpace，mapBase，mapSize，access，suggested，address）  1） 目标：内存空间映射  2） 参数表：  参数名 输入/输出方向   数据类型    描述  Vi 输入    ViSession   对话通道标识符（句柄）  MapSpace    输入    ViUInt16    映射地址空间  MapBase 输入    ViBusAddress    内存偏移量  MapSize 输入    ViBusSize   映射内存大小  Access 输入    ViBoolean   映射特性  Suggested   输入    ViBuf   映射是否允许确认  Address 输出    ViPBuf 映射地址  3） 返回状态值：  完成代码：  VI\_SUCCESS 地址映射完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_SPACE 地址空间不当  VI\_ERROR\_INV\_OFFSET 偏移量不当  VI\_ERROR\_NSUP\_BYTE\_ORDER    不支持特定字节顺序  VI\_ERROR\_NSUP\_OFFSET    硬件不支持特定地址偏移量  VI\_ERROR\_NSUP\_PRIV 硬件不支持存取特性  VI\_ERROR\_TMO    超时错误  4） 描述：该操作进行特定内存空间映射。  5） 相关项：参见viUnmap（）。  6） 实现要求：在由于资源限制致使viMap（）不能立即执行，则其将沉睡到资源有效或超时情况发生。  4.2.    viUnMap（vi）  1） 目标：取消内存映射  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  3） 返回状态值：  完成代码：  VI\_SUCCESS 映射取消完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_WINDOW\_NMAPPED 标识符不当  4） 描述：该操作应与映射操作一一对应。  5） 相关项：参见viMap（）。  6） 实现要求：无。  4.3.    viPoke（vi，addr，width，value）、viPoke8（vi，addr，value）      viPoke16（vi，addr，value）、viPoke32（vi，addr，value）      viPoke64（vi，addr，value）  1） 目标：写数据  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  addr    输入    ViBuf   数据存储地址  width   输入    ViUInt16    数据长度  value   输入    ViBuf、ViUInt8、ViUInt16、ViUInt32、ViUInt64    存储数据值  3） 返回状态值：无  4） 描述：该操作将数据写入到addr参数表示的地址中。  5） 相关项：参见viPeek（）。  6） 实现要求：数据类型viBuf对应viPoke（），viUInt8对应viPoke8（），viUInt16对应viPoke16（），viUInt32对应viPoke32（），viUInt64对应viPoke64（）。  4.4.    viPeek（vi，addr，width，value）、viPeek8（vi，addr，value）      viPeek16（vi，addr，value）、viPeek32（vi，addr，value）      viPeek64（vi，addr，value）  1） 目标：读数据  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  addr    输入    ViBuf   数据存储地址  width   输入    ViUInt16    数据长度  value   输出    ViBuf、ViUInt8、ViUInt16、ViUInt32、ViUInt64    存储数据值  4） 描述：该操作从addr参数表示的地址中读取数据。  5） 相关项：参见viPoke（）。  6） 实现要求：其中数据类型viBuf对应viPeek（），viPUInt8对应viPeek8（），viPUInt16对应viPeek16（），viPUInt32对应viPeek32（），viPUInt64对应viPeek64（）。  n   器件特定命令资源（VI\_RSRC\_DEV\_CMD）：  1. 资源概述：允许发送与接收命令，并取得命令请求与命令反应，在检查到协议错误时产生一个协议错误信息。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_HW\_EN   硬件允许与否  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  VI\_ATTR\_PHYS\_ADDR   器件地址  3. 器件特定命令资源定义的事件如下所示：  事件    说明  VI\_EVENT\_DEV\_CMD    接收到命令或查询事件  4. 器件特定命令资源定义的操作如下：  viSendCommand（vi，command，param）  viQueryRequest（vi，operationMode，command，param，len，response）  viGenProtError（vi，protError）  4.1.    viSendCommand（vi，command，param）  1） 目标：向器件发送命令  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  command 输入    ViUInt32    命令列表  param   输入    ViAddr 参数值  3） 返回状态值：  完成代码：  VI\_SUCCESS 命令发送完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_BERR   总线错误  VI\_ERROR\_RAW\_RD\_PROT\_VIOL读协议错误  VI\_ERROR\_RAW\_WR\_PROT\_VIOL写协议错误  VI\_ERROR\_OUTP\_PROT\_VIOL 输出协议错误  VI\_ERROR\_INP\_PROT\_VIOL 输入协议错误  4） 描述：该操作向器件发送特定命令，命令包括：获取硬件版本信息、获取软件版本信息、发送字串命令、发送长字串命令、发送扩展字串命令、发这GPIB命令及发送接口缺省命令。  5） 相关项：参见viQueryRequest（）。  6） 实现要求：无。 |

写资源

|  |
| --- |
| 图3.4 VISA仪器控制资源关系概图  在VISA仪器控制资源中，CPU接口资源是相对独立的，它与其它仪器控制资源无内部联系。这并不是说CPU接口资源不能与其它资源相互调用，但它不能调用其它仪器控制资源的相关操作，也不能被其它仪器控制资源调用其相关操作。而如格式化I/O资源的操作是依赖于写资源及读资源操作的，它们之间存在着内部关联性。同样，触发资源、状态/服务请求资源及清除资源与器件特定命令资源，高级存取资源与低级存取资源之间也存在着内部关联性。  n   写资源（VI\_RSRC\_WR）：  1. 资源概述：控制器向器件传送任意数据块，器件可以将接收到的数据块解释为消息、命令或二进制编码数据。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_FDC\_ACCESS\_MODE FDC(高速数据通道)缓冲区存取模式  VI\_ATTR\_FDC1\_SUPP   FDC版本1是否有效  VI\_ATTR\_FDC2\_SUPP   FDC版本2是否有效  VI\_ATTR\_FDC\_CHNL    FDC数据传送通道  VI\_ATTR\_FDC\_GEN\_SIGNAL\_EN   是否允许通过FDC传送数据  VI\_ATTR\_FDC\_MODE    FDC模式  VI\_ATTR\_FDC\_USE\_PAIR    一对或一个FDC有效  VI\_ATTR\_FDC\_USE\_VER FDC协议版本  VI\_ATTR\_INP\_RDY\_NOTIFY\_EN   数据有效是否被通知  VI\_ATTR\_NRDY\_ABORT\_EN   器件不再接收数据时是否被通知  VI\_ATTR\_PHYS\_ADDR   器件地址  VI\_ATTR\_REPEAT\_ADDR\_EN 是否使用重复地址  VI\_ATTR\_SEND\_END\_ON 终止符是否被确认  \_TERMCHAR\_EN  VI\_ATTR\_SEND\_END\_ON 是否将END作为终止符  VI\_ATTR\_TERMCHAR    终止符  VI\_ATTR\_TMO\_UNIT    超时值单位  VI\_ATTR\_TMO\_VALUE   超时值  VI\_ATTR\_TRANSFER\_MECH   传送机制(DMA或PIO)  VI\_ATTR\_WR\_PROT 传送协议  VI\_ATTR\_CMDR\_SERV   主从模式  VI\_ATTR\_ASRL\_BAUD   波特率  VI\_ATTR\_ASRL\_DATA\_BITS 数据位  VI\_ATTR\_ASRL\_PARITY 校验极性  VI\_ATTR\_ASRL\_STOP\_BITS 停止位  VI\_ATTR\_ASRL\_FLOW\_CNTRL 数据流控制状态  3. 写资源定义的事件如下所示：  事件    说明  VI\_EVENT\_IO\_COMPLETION 异步I/O操作已完成  4. 写资源定义的操作如下：  viWrite（vi，buf，count，retCount）  viWriteAsync（vi，buf，count，jobId）  viWriteStatus（vi，jobId，jobStatus）  4.1.    viWrite（vi，buf，count，retCount）  1） 目标：将数据同步写入到器件中  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  buf 输入    ViBuf   数据块位置  count   输入    ViUInt32    写入的字节数  retCount    输出    ViPUInt32   实际传送字节数  3） 返回状态值：  完成代码：  VI\_SUCCESS 同步写操作完成  VI\_WARN\_NRDY\_ABORT 同步写操作退出，器件未准备好输入数据  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_TMO    超时错误  VI\_ERROR\_RAW\_WR\_PROT    传送时写协议被破坏  \_VIOL  VI\_ERROR\_RAW\_RD\_PROT    传送时读协议被破坏  \_VIOL  VI\_ERROR\_OUTP\_PROT\_VIOL 传送时输出协议错误  VI\_ERROR\_INP\_PROT\_VIOL 传送时输入协议错误  VI\_ERROR\_BERR   传送时总线错误  VI\_ERROR\_ABORT 传送时用户退出  VI\_ERROR\_IN\_PROGRESS    同步写已执行  VI\_ERROR\_INV\_SETUP 设置错误，不能开始写操作  4） 描述：该操作将位于buf中的数据写入到器件中去，任一写资源通道只能登录一个同步写操作。  5） 相关项：参见viWriteAsync（）、viTerminate（）、viWriteStatus（）、viRead（）、viReadAsync（）。  6） 实现要求：无。  4.2.    viWriteAsync（vi，buf，count，jobId）  1） 目标：将数据异步写入到器件中  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  buf 输入    ViBuf   数据块位置  count   输入    ViUInt32    写入的字节数  jobId   输出    ViPjobId    异步写操作作业标识符  3） 返回状态值：  完成代码：  VI\_SUCCESS 异步写操作完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_QUEUE\_ERROR    不能进行写操作排队  4） 描述：该操作将位于buf中的数据异步写入到器件中去，任一写资源通道可以登录多个异步写操作，进行排队处理。  5） 相关项：参见viWrite（）、viTerminate（）、viWriteStatus（）、viRead（）、viReadAsync（）。  6） 实现要求：无。  4.3.    viWriteStatus（vi，jobId，jobStatus）  1） 目标：获取写操作状态  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  jobId   输入    VijobId 写操作作业标识符  jobStatus   输出    ViPJobstatus    写操作状态  3） 返回状态值：  完成代码：  VI\_SUCCESS 状态获取完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_JOBID 作业标识符不当  4） 描述：当操作为同步时，标识符唯一确定，为VI\_SYNC；当操作为异步时，标识符为操作的输出参数值。  5） 相关项：参见viWriteAsync（）。  6） 实现要求：无。 |

VISA仪器控制资源

|  |
| --- |
| 在VISA资源模板的基础上，VISA仪器控制资源定义了仪器控制特有的属性、事件与操作。在仪器控制资源中定义的功能是通用的仪器功能，它适用于各种仪器类型。  VISA仪器控制资源中包括：  n   写资源  n   读资源  n   格式化I/O资源  n   触发资源  n   状态/服务请求资源  n   清除资源  n   高级存取资源  n   低级存取资源  n   器件特定命令资源  n   CPU接口资源  VISA仪器控制各资源的关系如图3.4所示 |

**V**ISA资源定义

|  |
| --- |
| 下面对VISA资源描述格式中各部分进行逐一分析：  1、 资源概述：简要说明资源所要实现的功能，一般不针对特定仪器类型或接口进行描述，只提供一个通用概念。  2、 资源属性表及属性描述：资源属性表列出了资源所有属性，并在以后的属性描述中作较详细的说明。资源属性表如表3.3所示。  属性符号名 存取特性    数据类型    取值范围 读/写特性   私有/公有特性  表3.3 VISA资源属性表  l   属性符号名：用于标识属性。  l   存取特性：包括读/写特性与私有/公有特性。读/写特性可选择R/W（可读可写）与RO（只读）；私有/公有特性可选择Local（对话通道私有）与Global（资源内所有对话通道公有）。  l   数据类型：VISA数据类型是独立于编程语言定义的，在具体编程中调用VISA函数进行实际操作时，VISA数据类型与具体数据类型将一一映射。VISA定义了两类数据类型：基本数据类型与复杂数据类型。分别见表3.4和表3.5。  l   取值范围：定义了该属性数据的取值范围。  基本数据类型    数据类型描述    基本数据类型    数据类型描述  ViChar 字符类型    ViInt32 32位整型  ViPChar 字符指针类型    ViPInt32    32位整型指针类型  ViByte 字节类型    ViInt64 64位整型  ViPByte 字节指针类型    ViPInt64    64位整型指针类型  ViString    字符串类型 ViUInt8 8位无符号整型  ViPString   字符串指针类型 ViPUInt8    8位无符号整型指针类型  ViBoolean   布尔类型    ViUInt16    16位无符号整型  ViPBoolean 布尔指针类型    ViPUInt16   16位无符号整型指针类型  ViInt8 8位整型 ViUInt32    32位无符号整型  ViPInt8 8位整型指针类型 ViPUInt32   32位无符号整型指针类型  ViInt16 16位整型    ViUInt64    64位无符号整型  ViPInt16    16位整型指针类型    ViPUInt64   64位无符号整型指针类型  ViReal32    32位实型    ViReal64    64位实型  ViPReal32   32位实型指针类型    ViPReal64   64位实型指针类型  表3.4 VISA基本数据类型  复杂数据类型    数据类型描述    复杂数据类型    数据类型描述  ViRsrc 资源标识类型    ViHndlr 操作句柄类型  ViPRsrc ViRsrc指针类型 ViPHndlr    ViHndlr指针类型  ViStatus    返回状态值类型 ViVAList    参数列类型  ViPStatus   ViStatus指针类型    ViJobId 操作请求类型  ViBuf   数据块类型 ViPJobId    ViJobId指针类型  ViPBuf ViBuf指针类型   ViJobStatus 操作请求状态类型  ViAddr 逻辑地址类型    ViPJobStatus    ViJobStatus指针类型  ViPAddr ViAddr指针类型 ViSpaceInfo 内存映射类型  ViSession   资源对话通道类型    ViPhysAddr 物理地址类型  ViPSession ViSession指针类型   ViSigMask   信号过滤类型  ViVersion   资源版本类型    ViIntrMask 中断屏蔽类型  ViPVersion ViVersion指针类型   ViBusAddress    总线地址类型  ViObject    资源对象类型    ViBusSize   地址长度类型  ViPObject   ViObject指针类型    ViAttr 资源属性类型  ViRsrcList 资源标识列类型 ViAttrState 资源属性值类型  ViClass 资源类类型 ViLock 资源进程/线程管理类型  ViEvent 资源事件类型    ViPLock ViLock指针类型  ViPEvent    ViEvent指针类型 ViEventType 资源事件类型类型  ViAccessMode    控制存取机制类型    ViPEventType    ViEventType 指针类型  表3.5 VISA复杂数据类型  3、 资源事件集：VISA定义了一种运行机制，用于在一定条件下通知应用程序一个行为的发生，这些条件与发生被称为VISA事件。VISA事件模型如图3.2所示。  VISA事件模型由三部分组成。1）事件捕捉与通知：资源感知到事件发生，并与对话通道相联系；2）事件请求与处理：告知应用程序事件已发生，VISA定义了事件处理机制；3）事件响应：根据协议对事件进行响应。  图3.2 VISA事件模型  根据VISA事件引发的原因，事件类型可分为以下几种：  l   由硬件请求产生：如GPIB器件SRQ信号有效。  l   由硬件响应产生：如VXI总线中的SYSFAIL线有效。  l   由于资源行为的起始与结束所产生：如应用程序想知道系统服务是处于在线状态还是离线状态。  l   由于资源转换到不正常操作，必须终止正常操作所产生的事件。  l   在应用程序正常运行过程中的错误散转情况。  在VISA中，定义了两种事件处理机制。一种是事件排队方式（Queuing），它主要分两步进行：首先允许特定事件的处理机制，然后在一定时刻点去查询事件是否已经发生（调用VISA中的viWaitOnEvent()函数，事件类型见VI\_EVENT\_类参数），在事件发生之后，必须关闭特定事件的处理机制，以防止资源溢出。例3.4所示为用事件排队方式处理VISA事件。  例3.4 排队方式处理VISA事件  另一种事件处理方式为回调方式（Callback），也即中断方式。它首先安装回调句柄，然后在程序中允许事件处理机制，并应编写相应的中断子程序。利用回调方式进行处理的事件类型主要有以下几种：  l   VI\_EVENT\_SERVICE\_REQ：该事件表示VXI器件服务请求  l   VI\_EVENT\_TRIG：该事件表示从VXI器件接收到硬件触发信号  l   VI\_EVENT\_VXI\_SIGP：该事件表示从VXI器件接收到总线信号或总线中断信号  例3.5、例3.6与例3.7分别是利用回调方式对各种类型事件进行处理的实例，其中  例3.5是对HP公司推出的VXI数字万用表模块进行服务请求事件的处理，而例3.6与  例3.7是对浙江大学数字所自主开发的VXI数字输入/输出模块进行触发事件与总线中断事件处理。事实上，这几种事件处理示例可直接推广应用到其它的消息基器件与寄存器基器件中。  例3.5 利用回调方式对VXI模块进行服务请求事件处理：  例3.5 回调方式进行服务请求事件处理  在这种模式中，主程序安装的事件句柄类型为VI\_EVENT\_SERVICE\_REQ，并须将服务请求使能。当服务请求事件发生时，安装的事件句柄被激活，系统进入中断子程序。中断子程序首先对事件类型进行判断，然后进行具体事件处理。在主程序退回时，必须关闭事件允许机制，并将安装的事件句柄进行卸载。  例3.6 利用回调方式对VXI模块进行触发事件处理：  例3.6 回调方式进行触发事件处理  在这种模式中，主程序安装的事件句柄类型为VI\_EVENT\_TRIG，并须将服务请求使能。本主程序利用设置仪器属性的方式进行触发总线的选择与有效，并利用软件进行同步触发信号线TTL0。当触发事件发生时，安装的事件句柄被激活，系统进入中断子程序。中断子程序首先对事件类型进行判断，然后进行具体事件处理。需要注意的是，在中断子程序中获取事件的句柄不再是仪器句柄vi，而是ViEvent类型的事件句柄ctx。在主程序退回时，必须将安装的事件句柄进行卸载。  例3.7 利用回调方式对VXI模块进行总线中断事件处理：  例3.7 回调方式进行总线中断事件处理  在这种模式中，主程序安装的事件句柄类型为VI\_EVENT\_VXI\_SIGP，并须将服务请求使能。本主程序直接对器件内部控制寄存器进行写操作，允许中断产生并强制中断产生。当触发事件发生时，安装的事件句柄被激活，系统进入中断子程序。中断子程序首先对事件类型进行判断，然后进行具体事件处理。同样，在中断子程序中获取事件的句柄不再是仪器句柄vi，而是ViEvent类型的事件句柄ctx。在主程序退回时，必须将安装的事件句柄进行卸载。  4、 资源操作集：包括了一系列VISA资源操作。而每个操作又分别包括以下各项：  l   名字（含形参名）：用于操作函数原型的简单描述。  l   目标：简要描述相应的操作功能。  l   参数表：列出了操作中所有参数的相关信息，包括参数名、输入/输出方向、参数数据类型及参数描述。操作参数表如表3.6所示。  参数名 输入/输出方向   数据类型    描述  表3.6资源操作参数表  l   返回状态值：VISA资源操作返回状态值类型是唯一的，即为ViStatus。操作返回状态值包括完成代码与错误代码两类。  l   描述：对每个函数的功能及其中关键参数的含义等作较详细的描述。  l   相关操作项：可参考的其它相关操作。  l   实现要求：操作实现时应满足的一些条件与备注要求。  3.3 VISA资源定义  3.3.1 VISA资源模板  VISA资源模板相当于面向对象程序（OOP）中的基类，VISA其它资源类全部是它的子类。VISA资源模板上本身没有定义资源，它定义的是公共属性、事件与操作，所有VISA资源全部继承了资源模板的属性、事件与操作。在其它VISA资源定义中，实际上是定义了资源特有的属性、事件与操作，VISA资源真正拥有的属性、事件与操作包括资源模板定义的公共部分与资源特有的私有部分。  VISA资源模板中主要定义了以下几个共性功能：  l   资源定位与查寻  l   资源创建与删除  l   资源属性读取与修改  l   资源存取模式与权限控制  l   基本通讯服务（包括操作激活与事件报告）  VISA资源模板的定义也包括相应的属性、事件与操作，各自的描述格式与定义方式与其它VISA资源描述是一致的，这主要是为了其他资源可以方便地从基类继承所有属性、事件与操作。  资源模板定义的属性如表3.7所示：  属性符号名 存取特性    数据类型    取值范围      读写特性    公私特性  VI\_ATTR\_RSRC\_CLASS 只读    公有    ViClass 无  VI\_ATTR\_RSRC\_NAME   只读    公有    ViString    无  VI\_ATTR\_RSRC\_VERSION    只读    公有    ViVersion   无  VI\_ATTR\_RSRC\_LOCK\_STATE 只读    公有      ViUInt16    VI\_NO\_LOCKVI\_NON\_EXCLUSIVE\_LOCKVI\_EXCLUSIVE\_LOCK  VI\_ATTR\_MAX\_QUEUE\_LENGTH    可读可写    私有    ViUInt32    无  VI\_ATTR\_QUEUE\_TRIP\_POINT    可读可写    私有    ViUInt32    无  VI\_ATTR\_USER\_DATA   可读可写    私有    ViAddr 无  表3.7 VISA资源模板属性表  VISA资源模板属性描述：  属性名          描述  VI\_ATTR\_RSRC\_CLASS 资源所属类的标识符  VI\_ATTR\_RSRC\_NAME   资源名标识符  VI\_ATTR\_RSRC\_VERSION    资源版本标识符  VI\_ATTR\_RSRC\_LOCK\_STATE 资源存取锁定模式（取值为表3.7中所列三种）  VI\_ATTR\_MAX\_QUEUE\_LENGTH    任一时间特定对话通道最大排队的事件长度  VI\_ATTR\_QUEUE\_TRIP\_POINT    排队事件的触发点  VI\_ATTR\_USER\_DATA   资源特定对话通道所用的私有数据  资源模板定义的事件如下所示：  事件    说明  VI\_EVENT\_ALERT 通知应用程序对话通道或资源退出、复位或关闭  VI\_EVENT\_RSRC\_ACTIVE    通知应用程序资源已激活  VI\_EVENT\_RSRC\_INACTIVE 通知应用程序资源未被激活  VI\_EVENT\_RSRC\_LOCK\_CHANGED 通知应用程序存取锁定状态已更新  VI\_EVENT\_QUEUE\_FULL 通知应用程序等待操作事件队列已满  VI\_EVENT\_QUEUE\_TRIP 通知应用程序事件等待操作触发点已到  VI\_EVENT\_RSRC\_AVAILABLE 通知应用程序特定资源已有效  资源模板定义的操作如下：  viFindRsrc（vi，expr，accessLink）  viOpen（sesn，rsrcname，accessMode，timeout，vi）  viClose（vi）  viTerminate（vi，target，degree，jobId，immediate，flagFlushData）  viAttachRsrc（vi，rsrcName，refName）  viDetachRsrc（vi，refName）  viSetAttribute（vi，attribute，attrState）  viSetRsrcAttribute（vi，rsrcName，attribute，attrState）  viGetAttribute（vi，attribute，attrState）  viGetRsrcAttribute（vi，attribute，attrState）  viLock（vi，lockType，shareType，timeout，lockId）  viLockRsrc（vi，rsrcName，lockType，shareType，timeout，lockId）  viUnlock（vi，lockId）  viUnlockRsrc（vi，rsrcName，lockId）  viEnableEvent（vi，eventType，mechanism，context）  viDisableEvent（vi，eventType，mechanism）  viQueryEventMech（vi，eventType，mechanism）  viInstallHandler（vi，eventType，handler，userHandle）  viUnInstallHandler（vi，eventType，handler，userHandle）  viQueryHandlers（vi，eventType，handlers，userHandle）  viAcknowledgeEvent（vi，eventType，context）  viWaitOnEvent（vi，eventType，timeout，outContext）  viWaitOnMultipleEvent（vi，eventTypeList，timeout，outContext，outEventType）  viDiscardEvents（vi，eventType，mechanism）  viGetEventInfo（vi，eventType，context，rsrcName，info）  viRaiseEvent（vi，eventType，context，target）  viEventHandler（vi，eventType，context，userHandle）  下面将分别对各个操作作一详细描述。  1、 viFindRsrc（vi，expr，accessLink）  1） 目标：查询VISA系统，进行资源定位  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   资源对话通道标识符  expr    输入    ViString    用于资源名匹配的表达式  accessLink 输出    ViRsrcList 资源所在位置列表  3） 返回状态值：  完成代码：  VI\_SUCCESS 资源查寻到  错误代码：  VI\_ERROR\_INV\_SESSION    vi（句柄）不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EXPR   查找表达式不规范  VI\_ERROR\_RSRC\_NFOUND    查找表达式无法与资源名匹配  4） 描述：该操作用于查询资源名与表达式相匹配的资源，并确定资源位置。  5） 相关项：参见viAttachRsrc（）、viOpen（）。  6） 实现要求：输出参数accessLink应含所有相匹配的资源个数。  2、 viOpen（sesn，rsrcName，accessMode，timeout，vi）  1） 目标：打开特定资源的对话通道  2） 参数表：  参数名 输入/输出方向   数据类型    描述  sesn    输入    ViSession   VISA资源管理器  rsrcName    输入    ViRsrc 资源名  accessMode 输入    ViAccessMode    资源存取锁定模式  timeout 输入    ViUInt32    操作超时值  vi 输出    ViPSession 对话通道标识符（句柄）  3） 返回状态值：  完成代码：  VI\_SUCCESS 对话通道打开完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_INV\_RSRC\_NAME 资源名语法错误  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_ACC\_MODE   不合法的存取模式  VI\_ERROR\_NSUP\_ACC\_MODE 存取模式不被支持  VI\_ERROR\_RSRC\_NFOUND    系统中未能找到该资源  VI\_ERROR\_TMO    操作超时  4） 描述：该操作用于开启特定资源的一个对话通道，它返回的对话通道标识符可作为以后操作的软件句柄。参数中的accessMode应在几种存取模式中选取（VI\_NO\_LOCK、VI\_EXCLUSIVE\_LOCK、VI\_NON\_EXCLUSIVE\_LOCK）。  5） 相关项：参见viClose（）。  6） 实现要求：sesn参数一般用缺省VISA资源管理器viDefaultRM代入，因此在调用viOpen（）函数之前一般已先调用了viOpenDefaultRM（）函数用于打开缺省的资源管理器。  3、 viClose（vi）  1） 目标：关闭特定的对话通道  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  3） 返回状态值：  完成代码：  VI\_SUCCESS 对话通道关闭完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_CLOSE\_FAILED   无法释放与该对话通道相关联的内存数据结构  4） 描述：该操作用于关闭与资源相关联的对话通道，并释放内存数据。  5） 相关项：参见viOpen（）。  6） 实现要求：当一个对话通道在执行viClose（）操作时，不仅应关闭该通道，并应释放所有通道的数据。  4、 viAttachRsrc（vi，rsrcName，refName）  1） 目标：通过对话通道，建立两资源之间的关联  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  rsrcName    输入    ViRsrc 相关联的资源名  refName 输入/输出   ViPRsrc 相关联的资源别名  3） 返回状态值：  完成代码：  VI\_SUCCESS 资源关联完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_INV\_RSRC\_NAME 资源名语法错误  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_DUPLICATE refName参数中的资源已存在  \_REF\_NAME  VI\_ERROR\_ATTACH\_REFUSED 资源无法被关联  4） 描述：该操作用于关联两个资源，用vi标识的对话通道可以存取相关联资源的所有属性与操作。  5） 相关项：参见viDetachRsrc（）、viGetRsrcSession（）。  6） 实现要求：资源间应通过关联方式实现对话。  5、 viDetachRsrc（vi，refName）  1） 目标：取消资源之间的关联  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  refName 输入/输出   ViPRsrc 相关联的资源别名  3） 返回状态值：  完成代码：  VI\_SUCCESS 取消资源关联完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_RSRC\_NFOUND    资源未能找到  VI\_ERROR\_INV\_RSRC\_NAME 资源名语法错误  4） 描述：该操作用于取消资源之间关联，一旦关联取消，相关联资源的所有属性与操作不能再被存取。  5） 相关项：参见viAttachRsrc（）。  6） 实现要求：资源无法拒绝取消关联的请求。  6、 viTerminate（vi，target，degree，jobId，immediate，flagFlushData）  1） 目标：请求VISA资源终止一个或所有对话通道的正常运行  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  target 输入    ViBoolean   请求目标（资源或通道）  degree 输入    ViUInt16    请求类型，正常类型有VI\_ABORT、VI\_RESET、VI\_FAIL、  VI\_KILL  jobId   输入    ViJobId 操作标识符  immediate   输入    ViBoolean   请求是否立即有效  flagFlushData   输入    ViBoolean   资源是否放弃应用数据  3） 返回状态值：  完成代码：  VI\_SUCCESS 请求服务完成  VI\_SUCCESS\_STATE\_UNKNOWN    请求服务完成，但资源可能处于不一致状态  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_DEGREE 请求类型不合法  VI\_ERROR\_INV\_JOB\_ID 操作类型或标识符不合法  4） 描述：该操作用于请求资源去终止一个或所有对话通道的正常运行，请求类型可分为VI\_ABORT（退出）、VI\_RESET（复位）、VI\_FAIL（失败）及VI\_KILL（停止所有资源操作）四种。  5） 相关项：参见VI\_EVENT\_ALERT事件描述。  6） 实现要求：当VISA资源激活viTerminate（）操作时，资源不能进行其它操作。  7、 viGetAttribute（vi，attribute，attrState）          viGetRsrcAttribute（vi，rsrcName，attribute，attrState）  1） 目标：获取资源属性状态值  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  rsrcName    输入    ViRsrc 资源名  attribute   输入    ViAttr 资源属性  attrState   输出    ViPAttrState    资源属性状态值  3） 返回状态值：  完成代码：  VI\_SUCCESS 获取属性完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_INV\_RSRC\_NAME 资源名语法错误  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_NSUP\_ATTR 资源属性未定义  4） 描述：viGetAttribute（）取回当前资源属性状态值，而viGetRsrcAttribute（）取回其它资源的公有属性状态值。  5） 相关项：参见viSetAttribute（）、viSetrsrcAttribute（）、viLock（）、viUnLock（）。  6） 实现要求：无论资源锁定状态如何，viGetRsrcAttribute（）操作都可以取回资源公有属性状态值。  8、 viSetAttribute（vi，attribute，attrState）      viSetRsrcAttribute（vi，rsrcName，attribute，attrState）  1） 目标：设置资源属性状态值  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  rsrcName    输入    ViRsrc 资源名  attribute   输入    ViAttr 资源属性  attrState   输入    ViAttrState 资源属性状态值  3） 返回状态值：  完成代码：  VI\_SUCCESS 属性设置完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_INV\_RSRC\_NAME 资源名语法错误  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_NSUP\_ATTR 资源属性未定义  VI\_ERROR\_NSUP\_ATTR\_STATE    资源属性状态值不支持  VI\_ERROR\_NSUP\_ATTR 资源属性为只读状态  \_READONLY  4） 描述：viSetAttribute（）设置当前资源属性状态值，而viSetRsrcAttribute（）设置其它资源的公有属性状态值。  5） 相关项：参见viGetAttribute（）、viGetrsrcAttribute（）、viLock（）、viUnLock（）。  6） 实现要求：当资源处于锁定状态时，无法进行属性状态值设置。  9、 viLock（vi，lockType，shareType，timeout，lockId）          viLockRsrc（vi，rsrcName，lockType，shareType，timeout，lockId）  1） 目标：设置资源存取模式  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  rsrcName    输入    ViRsrc 资源名  lockType    输入    ViUInt16    锁定类型（独占或非独占）  shareType   输入    ViUInt16    锁定共享类型  timeout 输入    ViUInt32    操作超时值  lockId 输入/输出   ViPLock 当锁定设置成功后，资源返回的唯一锁定标识符  3） 返回状态值：  完成代码：  VI\_SUCCESS 锁定模式设置完成  VI\_SUCCESS\_SHARE\_LOCK   锁定模式设置完成，并被多个对话通道共享  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_INV\_RSRC\_NAME 资源名语法错误  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_LOCK\_TYPE 锁定模式不当  VI\_ERROR\_INV\_LOCK\_ID    锁定标识符不当  VI\_ERROR\_NSHARABLE lockID不能用于非共享模式  VI\_ERROR\_LOCK\_CHANGE    由于对话通道已被锁定，不能从非独占锁定状  \_FAILED 态改变到独占状态  4） 描述：该操作用于设置资源锁定模式，必须区分锁定类型与共享类型。共享类型的设置只对于独占锁定状态有关，所有非独占性锁定状态均为共享。  5） 相关项：参见viUnLock（）、viUnLockRsrc（）。  6） 实现要求：当锁定模式为非独占型时，shareType参数可忽略。  10、    viUnLock（vi，lockId）          viUnLockRsrc（vi，rsrcName，lockId）  1） 目标：取消资源存取模式  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  rsrcName    输入    ViRsrc 资源名  lockId 输入    ViLock 锁定标识符  3） 返回状态值：  完成代码：  VI\_SUCCESS 取消锁定模式完成  VI\_WARN\_NON\_EXCLUSIVE   调用成功，但资源仍被非独占型锁定  \_LOCK  VI\_WARN\_EXCLUSIVE\_LOCK 调用成功，但资源仍被独占型锁定  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_INV\_LOCK\_ID    锁定标识符不当  VI\_ERROR\_LOCK\_NOWNER    锁定并非来自于特定通道vi  4） 描述：该操作用于取消以前设置的资源锁定模式。  5） 相关项：参见viLock（）、viLockRsrc（）。  6） 实现要求：注意不同的参数输入对应于不同的返回状态。  11、    viEnableEvent（vi，eventType，mechanism，context）  1） 目标：允许特定事件通知  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  mechanism   输入    ViUInt16    事件处理机制  context 输入    ViEvent 事件信息  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件通知允许  VI\_SUCCESS\_EVENT\_EN 至少一种事件处理机制被允许  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_MECH   事件处理机制不合法  VI\_ERROR\_HNDLR\_NINSTALLED   特定事件句柄未被安装  4） 描述：该操作允许特定事件的通知，应用程序可采用事件排队或回调（中断）方式处理该事件。  5） 相关项：参见viEventHandler（）、viInstallHandler（）、viUnInstallHandler（）。  6） 实现要求：若事件句柄未被安装，则在回调（中断）方式中返回错误值。  12、    viDisableEvent（vi，eventType，mechanism）  1） 目标：不允许特定事件通知  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  mechanism   输入    ViUInt16    事件处理机制  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件通知不允许  VI\_SUCCESS\_EVENT\_DIS    至少一种事件处理机制不被允许错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_MECH   事件处理机制不合法  4） 描述：该操作取消特定事件的服务通知。  5） 相关项：参见viEventHandler（）、viInstallHandler（）、viUnInstallHandler（）。  6） 实现要求：若一个对话通道上的事件处理机制不允许，事件登录还将保留。  13、    viQueryEventMech（vi，eventType，mechanism）  1） 目标：查询事件处理机制  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  mechanism   输出    ViUInt16    事件处理机制  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件查询完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  4） 描述：该操作查询特定事件处理机制，所有事件处理机制返回值以“位或”操作值形式给出。  5） 相关项：参见viEventHandler（）、viInstallHandler（）、viUnInstallHandler（）。  6） 实现要求：无。  14、    viInstallHandler（vi，eventType，handler，userHandle）  1） 目标：安装事件回调句柄  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  handler 输入    ViHndlr 应用程序中的事件句柄  userHandle 输入    ViAddr 事件句柄值  3） 返回状态值：  完成代码：  VI\_SUCCESS 安装事件句柄完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  4） 描述：该操作允许应用程序去安装事件处理句柄，句柄由输入参数handler唯一确定，事件类型包括资源事件（VI\_RSRC\_EVENT）、系统事件(VI\_SYS\_EVENT)、异常事件（VI\_EXCEPTION），当事件类型参数设为所有事件(VI\_ALL\_EVENTS)时，以上三种类型事件句柄均被设置。  5） 相关项：参见viEventHandler（）、viUnInstallHandler（）。  6） 实现要求：无。  15、    viUninstallHandler（vi，eventType，handler，userHandle）  1） 目标：卸载事件回调句柄  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  handler 输入    ViHndlr 应用程序中的事件句柄  userhandle 输入    ViAddr 事件句柄值  3） 返回状态值：  完成代码：  VI\_SUCCESS 卸载事件句柄完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_HNDLR\_REF 句柄参数与所有已安装的句柄不匹配  4） 描述：该操作允许应用程序去卸载事件处理句柄，句柄由输入参数handler唯一确定，事件类型包括资源事件（VI\_RSRC\_EVENT）、系统事件(VI\_SYS\_EVENT)、异常事件（VI\_EXCEPTION），当事件类型参数设为所有事件(VI\_ALL\_EVENTS)时，以上三种类型事件句柄均被卸载。  4） 相关项：参见viEventHandler（）、viEnableEvent（）。  5） 实现要求：卸装的句柄一定要先安装。  16、    viQueryHandlers（vi，eventType，handlers，userHandle）  1） 目标：查询事件已安装句柄  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  handlers    输出    ViPHndlr    应用程序中的事件句柄  userHandle 输出    ViAddr 事件句柄值  3） 返回状态值：  完成代码：  VI\_SUCCESS 查询事件句柄完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  4） 描述：该操作查询一种事件类型的所有已安装的事件句柄。  5） 相关项：参见viInstallHandler（）、viUnInstallHandler（）。  6） 实现要求：无。  17、    viAcknowledgeEvent（vi，eventType，context）  1） 目标：对事件通知作响应  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  context 输入    ViEvent 事件信息  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件响应完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_CONTEXT    事件信息不合法  4） 描述：该操作响应已经报告的事件发生，事件发生由信息结构唯一确定。  5） 相关项：参见viEventHandler（）、viInstallHandler（）。  6） 实现要求：资源将在接收到所有通知后才对事件作出响应。  18、    viWaitOnEvent（vi，eventType，timeout，outContext）      viWaitOnMultipleEvents（vi，eventTypeList，timeout，outContext，  outEventType）  1） 目标：等待特定事件的发生  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViPEventType    事件标识符  eventTypeList   输入    ViPUInt16   事件标识符列表  timeout 输入    ViUInt32    操作超时值  outContext 输出    ViPEvent    事件信息  outEventType    输出    ViEventType 返回事件类型  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件等待完成  VI\_SUCCESS\_QUEUE\_EMPTY 事件等待完成，事件队列空  VI\_WARN\_QUEUE\_TRIP 事件等待完成。队列中有多个事件  VI\_WARN\_QUEUE\_FULL 事件等待完成。队列已满  VI\_WARN\_QUEUE\_OVERFLOW 事件等待完成。队列溢出且有事件丢失  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_CONTEXT    事件信息不合法  VI\_ERROR\_TMO    操作超时  VI\_ERROR\_QUEUE\_TRIP 队列中事件过多  VI\_ERROR\_QUEUE\_FULL 队列满  VI\_ERROR\_QUEUE\_OVERFLOW 队列事件溢出  4） 描述：viWaitOnEvent（）等待一个事件，viWaitOnMultipleEvents（）等待多个事件。  5） 相关项：参见viEnableEvent（）。  6） 实现要求：当超时值为VI\_INFINITE时，则会无限制等待下去。  19、    viDiscardEvents（vi，eventType，mechanism）  1） 目标：刷新一个对话通道上的事件发生  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  mechanism   输入    ViUInt16    事件处理机制  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件刷新完成  VI\_SUCCESS\_QUEUE\_EMPTY 事件刷新完成，事件队列空  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_MECH   事件处理机制不当  4） 描述：该操作刷新队列中所有事件发生与事件登录，所有未被处理事件的信息也将被丢掉。  5） 相关项：参见viEnableEvent（）。  6） 实现要求：该操作会引起事件丢失。  20、    viGetEventInfo（vi，eventType，context，rsrcName，info）  1） 目标：获取前一个事件发生的信息  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  context 输入/输出   ViEvent 事件信息  rsrcName    输出    ViString    资源名  info    输出    ViAddr 异常事件信息  3） 返回状态值：  完成代码：  VI\_SUCCESS 命令传递完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  4） 描述：该操作用于取回事件处理信息，返回信息包括一个事件信息数据结构。  5） 相关项：参见viRaiseEvent（）。  6） 实现要求：无。  21、    viRaiseEvent（vi，eventType，context，target）  1） 目标：通知资源或对话通道事件已发生  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  context 输入    ViEvent 事件信息  target 输入    ViBoolean   请求目标（资源或通道）  3） 返回状态值：  完成代码：  VI\_SUCCESS 通知事件发生完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_INV\_TARGET 请求目标不当  VI\_ERROR\_INV\_EVENT 特定事件不存在或不被支持  VI\_ERROR\_INV\_CONTEXT    事件信息不当  VI\_ERROR\_QUEUE\_FULL 事件队列满  VI\_ERROR\_NENABLED   当前无法感知事件  4） 描述：该操作用于通知资源或通道事件已发生，可以生成除了异常事件之外的所有类型事件。  5） 相关项：参见viEnableEvent（）、viWaitOnEvent。  6） 实现要求：该操作可用于模拟一个事件。  22、    viEventHandler（vi，eventType，context，userHandle）  1） 目标：事件服务过程原型  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  eventType   输入    ViEventType 事件标识符  context 输入    ViEvent 事件信息  userHandle 输入    ViAddr 事件句柄值  3） 返回状态值：  完成代码：  VI\_SUCCESS 事件处理完成  错误代码：  VI\_ERROR\_ABORT 终止操作引发异常  VI\_ERROR\_RESET 终止所有操作，复位至缺省状态  VI\_ERROR\_FAIL   终止对话通道所有操作  VI\_ERROR\_KILL   终止对话通道所有操作，并自动关闭通道  4） 描述：该操作在通道接收到一个事件并处理该事件时被调用。  5） 相关项：无。  6） 实现要求：在多个句柄情况下，只有所有句柄运行都终止时，正常运行才终止。  3.3.2 VISA资源管理器资源  VISA资源管理器（VI\_RSRC\_VISA\_RM）是VISA各资源的中间调度器，其在VISA各子系统元件在整个系统中配合工作中起着重要作用。资源管理器的基本功能包括：  1） 分配资源地址  2） 分配资源识别号（ID）  3） 进行操作调用  4） 进行事件管理  图3.3体现了VISA资源管理器在系统配置中的中间调度作用。  图3.3 VISA资源管理器与其它资源的关系示图  根据VISA规范，VISA资源管理器资源继承了资源模板定义的所有属性、事件与操作，在资源模板基础上本身没有加以扩展自己的属性、事件与操作。但资源管理器资源具有自己的语义，在系统资源中起着根资源的作用 |

VISA应用实例

|  |
| --- |
| 下面通过分别调用非VISA的I/O接口软件与VISA函数，对GPIB器件与VXI消息基器件进行简单的读/写操作（主要完成向器件发送查询器件标识符命令，并从器件读回响应值操作过程），进行VISA与其它I/O接口软件的异同点比较。所有例子中采用的编程语言均为C语言。  例3.1 用非VISA的I/O接口软件（NI公司的NI-488）实现对GPIB器件的读/写操作：  程序说明：  1、 声明区：声明程序中所有变量的数据类型，用C语言数据类型声明。  2、 开启区：进行GPIB器件初始化，确定GPIB器件地址，并为每个器件返回一个对应的器件句柄。在初始化过程中，器件句柄作为器件的标志以输出参数形式被返回，在其它的I/O功能函数中，器件句柄则作为函数的输入参数，以标志特定的GPIB器件。  3、 器件I/O区：在本例程中，器件I/O主要完成命令发送，并从GPIB器件中读回响应数据。主要完成向器件发送查询器件标识符命令，用标准命令符“\*IDN？”，并从器件读回响应值，即器件的标识符。在NI-488函数集中，分别用ibwrt函数与ibrd函数进行器件写操作与读操作。  4、 关闭区：GPIB的I/O软件将本身的数据结构存于内存中，当系统关闭时，所有仪器全部自动关闭，无需对I/O软件本身作关闭操作。也就是说，GPIB的I/O软件（如NI-488）无关闭机制。  例3.1 用NI-488实现对GPIB仪器的读/写操作  例3.2 用非VISA的I/O接口软件（NI公司的NI-VXI）实现对VXI消息基器件的读/写操作：  例3.2 用NI-VXI实现对GPIB器件的读/写操作程序说明：  1、 声明区：声明程序中所有变量的数据类型，用C语言数据类型声明。  2、 开启区：进行VXI消息基器件初始化，确定VXI消息基器件的逻辑地址。在对VXI器件操作中，逻辑地址取代了GPIB器件操作中的器件句柄，作为器件操作的标志，在初始化操作中返回唯一的值。  3、 器件I/O区：在本例程中，主要完成对命令发送，并从VXI消息基器件读回响应数据。同样完成向器件发送查询器件标识符命令，用标准命令符“\*IDN？”，并从器件读回响应值，即器件的标识符。由初始化得到的器件逻辑地址在器件I/O操作中作为函数的输入参数被使用。程序通过对逻辑地址的处理，完成对器件的一对一操作。在VXI消息基器件的操作中，分别用WSwrt函数与WSrd函数进行器件的写操作与读操作，其中这两个函数中的mode参数均表示数据传输方式，retCount参数表示实际传送的字节数。  4、 关闭区：对于VXI器件，存在着一个关闭机制，要求在结束器件操作的时候，同时关闭I/O接口软件。  例3.3 用VISA I/O接口软件实现对GPIB器件与VXI消息基器件的读/写操作：  例3.3 用VISA实现对GPIB器件与VXI消息基器件的读/写操作  程序说明：  1、 声明区：声明程序中所有变量的数据类型。与以上两例不同的是，在这儿声明的数据类型均为VISA数据类型，其是与编程语言无关的。而VISA数据类型与编程语言数据类型的对应说明，均包含在特定文件中。如VISA数据类型的C语言形式的包含头文件为本章附录文件visatype.h和visa.h。由于程序中没有涉及到具体某种语言的数据类型，故程序本身具有良好的兼容性与可移植性，各种编程语言调用VISA的数据类型与操作函数的参数调用格式相差甚少。  2、 开启区：进行消息基器件初始化，建立资源管理器及器件与VISA的通信关系。对所有器件进行初始化均调用viOpenDefaultRM()与viOpen()函数。在此例中，对于GPIB器件的初始化与对于VXI器件的初始化的函数调用形式是一致的，只是输入参数中器件描述符的值区别。在调用viOpen()函数时，器件硬件接口形式（计算机结构形式）是无需特别说明的，该初始化过程完全适用于各种器件硬件接口类型。初始化过程中返回的vi参数，类似于器件句柄，可作为器件操作的标志与数据传递的中介。  3、 器件I/O区：在本例程中，主要完成对器件发送命令，并从器件读回响应数据。对于GPIB的读/写操作与对于VXI消息基器件的读/写操作，调用的VISA函数是一样的。其中vi是操作函数的输入参数。  4、 关闭区：在器件操作结束时，调用viClose()函数，关闭器件及资源管理器与VISA的关联。  通过以上三个例程的分析，可以发现两个事实：  第一、VISA函数的调用与其它接口软件函数的调用形式上并无太多不同，学习功能强大的VISA软件并不比一般的I/O接口软件任务重。而且VISA的函数参数意义明确，结构一致，在理解与应用仪器程序时，效率较高。  第二、VISA用户只需学习了VISA函数应用格式，就可以对多种仪器实现统一控制，不必再象以前学会了用NI-488对GPIB器件操作之后，还得学会NI-VXI对VXI器件进行操作。与其它的I/O接口软件相比，VISA体现的多种结构与类型的统一性，使不同仪器软件可以运行在同一平台上，为虚拟仪器系统软件结构提供了坚实的基础。  3.2 VISA资源描述  3.2.1 VISA资源类与资源  自底向上的VISA模型内部是由面向对象程序（OOP）软件模块构成的。在VISA中，最基本的软件模块是定义在资源类上的资源。  VISA的资源类概念类似于面向对象程序设计方法中类的概念。类是一个实例外观和行为的描述。类通过构造函数来建立新的实例，因为不管怎样，先得建立一个实例才能使用它、操作它。当向类提出要建立一个实例的要求时，它会通过一个特定的构造函数创建并初始化一个新的实例。然而，类并不对销毁一个实例负责，这就要求实例要自己负责，它们通过调用一个方法来申请自我的销毁。类的声明定义了类的实例结构以及一组可以由该实例执行的有效操作。VISA资源类是一种抽象化的器件特点功能描述，是对资源精确描述的专用术语。  VISA的资源概念类似于面向对象程序设计方法中对象的概念。对象实例不仅包含数据实体，而且是一个服务提供者。做为一个数据实体，一个对象很象一个记录，由一些相同或不同类型的域构成。这些域的整体被称为一个对象的状态。改变这些域的值，逻辑上讲就是改变一个对象的状态。作为服务提供者，或者说是一个过程实体，一个对象是一些在其域上进行操作的方法或子程序的集合。从这个角度看来，它有点象一个子例程库。方法定义了对象的行为，激活一个对象方法的实际结果就是让对象做一个动作。这两种互相补充的角色使对象这种概念变得非常强大。使用对象，可以使抽象非常方便：一个对象的内部可以看成一个“黑匣子”，只能通过精心设计的由方法组成的接口来访问。对象的可见部分成为它的协议。  VISA中资源与OOP对象一样，也由三个要素组成：属性集、事件集与操作集。以读资源为例，其属性集包括结束字符串、超时值及协议等，事件集包括用户退出事件，操作集包括各种端口读取操作。  3.2.2 VISA资源描述格式  VISA资源是独立于编程语言与操作系统的，在VISA本身的资源定义与描述中并不包含任何操作系统或编程语言相关的限制。VISA源代码是唯一的，只为不同的操作系统编程语言提供了不同的API接口。VISA资源类共分为五大类：VISA资源模板、VISA资源管理器、VISA仪器控制资源、VISA仪器控制组织器、VISA特定接口仪器控制资源。在每一类中定义与描述的VISA资源都遵循同样的格式。VISA资源描述格式  如表3.1所示。  VISA资源描述格式是一种抽象定义，与具体编程语言无关，资源内所有元件的定义也均与编程语言无关。VISA通过提供不同的API接口，适用于不同的操作系统与编程环境。在不同的编程语言环境之中调用VISA，均需在应用程序头部引入说明文件。在C语言环境下，VISA资源说明文件为visatype.h和visa.h文件(详见本章附录)。唯一的VISA源程序通过不同的引入接口与文件说明，实现了不同环境下的适用性。VISA资源描述格式不仅适用于现在VISA包含的所有资源，也为VISA将来资源扩充定义了一个标准格式。现定义的VISA资源类型定义如表3.2所示。  X.1 资源概述  X.2 资源属性表及属性描述  X.3 资源事件集  X.4 资源操作集     所含每个操作包括：  X.4.Y   名字（含形参名）  X.4.Y.1 目标  X.4.Y.2 参数表  X.4.Y.3 返回状态值  X.4.Y.4 描述  X.4.Y.5 相关操作项  X.4.Y.6 实现要求  表3.1 VISA资源描述格式  资源    缩写名 标准名  VISA资源管理器资源 VRM VI\_RSRC\_VISA\_RM  VISA仪器控制组织器资源 VICO    VI\_RSRC\_VISA\_IC\_ORG  写资源 WR VI\_RSRC\_WR  读资源 RD VI\_RSRC\_RD  格式化I/O资源   FIO VI\_RSRC\_FMT\_IO  触发资源    TRIG    VI\_RSRC\_TRIG  清除资源    CLR VI\_RSRC\_CLR  状态/服务请求资源   SRQ VI\_RSRC\_SRQ  高级存取资源    HILA    VI\_RSRC\_HL\_ACC  低级存取资源    LOLA    VI\_RSRC\_LL\_ACC  器件特定命令资源    DEVC    VI\_RSRC\_DEV\_CMD  CPU接口资源 CPUI    VI\_RSRC\_CPU\_INTF  GPIB总线接口控制资源    GBIC    VI\_RSRC\_GPIB\_INTF  VXI总线器件配置资源 VXDC    VI\_RSRC\_VXI\_DEV\_CONF  VXI总线接口控制资源 VXIC    VI\_RSRC\_VXI\_INTF  VXI总线零槽资源 VXS0    VI\_RSRC\_VXI\_SLOT\_0  VXI总线系统中断资源 VXSI    VI\_RSRC\_SYS\_INTR  VXI总线信号处理器资源   VXSP    VI\_RSRC\_SIG\_PROCESSOR  VXI总线信号资源 VXS VI\_RSRC\_VXI\_SIG  VXI总线中断资源 VXIN    VI\_RSRC\_VXI\_INTR  VXI总线扩展器接口资源   VXEI    VI\_RSRC\_VXI\_EXTDR  异步串行总线接口控制资源    ASIC    VI\_RSRC\_ASRL\_INTF  表3.2 VISA资源类型定义  下面对VISA资源描述格式中各部分进行逐一分析：  1、 资源概述：简要说明资源所要实现的功能，一般不针对特定仪器类型或接口进行描述，只提供一个通用概念。  2、 资源属性表及属性描述：资源属性表列出了资源所有属性，并在以后的属性描述中作较详细的说明。资源属性表如表3.3所示。  属性符号名 存取特性    数据类型    取值范围 读/写特性   私有/公有特性  表3.3 VISA资源属性表  l   属性符号名：用于标识属性。  l   存取特性：包括读/写特性与私有/公有特性。读/写特性可选择R/W（可读可写）与RO（只读）；私有/公有特性可选择Local（对话通道私有）与Global（资源内所有对话通道公有）。  l   数据类型：VISA数据类型是独立于编程语言定义的，在具体编程中调用VISA函数进行实际操作时，VISA数据类型与具体数据类型将一一映射。VISA定义了两类数据类型：基本数据类型与复杂数据类型。分别见表3.4和表3.5。  l   取值范围：定义了该属性数据的取值范围。  基本数据类型    数据类型描述    基本数据类型    数据类型描述  ViChar 字符类型    ViInt32 32位整型  ViPChar 字符指针类型    ViPInt32    32位整型指针类型  ViByte 字节类型    ViInt64 64位整型  ViPByte 字节指针类型    ViPInt64    64位整型指针类型  ViString    字符串类型 ViUInt8 8位无符号整型  ViPString   字符串指针类型 ViPUInt8    8位无符号整型指针类型  ViBoolean   布尔类型    ViUInt16    16位无符号整型  ViPBoolean 布尔指针类型    ViPUInt16   16位无符号整型指针类型  ViInt8 8位整型 ViUInt32    32位无符号整型  ViPInt8 8位整型指针类型 ViPUInt32   32位无符号整型指针类型  ViInt16 16位整型    ViUInt64    64位无符号整型  ViPInt16    16位整型指针类型    ViPUInt64   64位无符号整型指针类型  ViReal32    32位实型    ViReal64    64位实型  ViPReal32   32位实型指针类型    ViPReal64   64位实型指针类型  表3.4 VISA基本数据类型  复杂数据类型    数据类型描述    复杂数据类型    数据类型描述  ViRsrc 资源标识类型    ViHndlr 操作句柄类型  ViPRsrc ViRsrc指针类型 ViPHndlr    ViHndlr指针类型  ViStatus    返回状态值类型 ViVAList    参数列类型  ViPStatus   ViStatus指针类型    ViJobId 操作请求类型  ViBuf   数据块类型 ViPJobId    ViJobId指针类型  ViPBuf ViBuf指针类型   ViJobStatus 操作请求状态类型  ViAddr 逻辑地址类型    ViPJobStatus    ViJobStatus指针类型  ViPAddr ViAddr指针类型 ViSpaceInfo 内存映射类型  ViSession   资源对话通道类型    ViPhysAddr 物理地址类型  ViPSession ViSession指针类型   ViSigMask   信号过滤类型  ViVersion   资源版本类型    ViIntrMask 中断屏蔽类型  ViPVersion ViVersion指针类型   ViBusAddress    总线地址类型  ViObject    资源对象类型    ViBusSize   地址长度类型  ViPObject   ViObject指针类型    ViAttr 资源属性类型  ViRsrcList 资源标识列类型 ViAttrState 资源属性值类型  ViClass 资源类类型 ViLock 资源进程/线程管理类型  ViEvent 资源事件类型    ViPLock ViLock指针类型  ViPEvent    ViEvent指针类型 ViEventType 资源事件类型类型  ViAccessMode    控制存取机制类型    ViPEventType    ViEventType 指针类型  表3.5 VISA复杂数据类型  3、 资源事件集：VISA定义了一种运行机制，用于在一定条件下通知应用程序一个行为的发生，这些条件与发生被称为VISA事件。VISA事件模型如图3.2所示。  VISA事件模型由三部分组成。1）事件捕捉与通知：资源感知到事件发生，并与对话通道相联系；2）事件请求与处理：告知应用程序事件已发生，VISA定义了事件处理机制；3）事件响应：根据协议对事件进行响应。  图3.2 VISA事件模型  根据VISA事件引发的原因，事件类型可分为以下几种：  l   由硬件请求产生：如GPIB器件SRQ信号有效。  l   由硬件响应产生：如VXI总线中的SYSFAIL线有效。  l   由于资源行为的起始与结束所产生：如应用程序想知道系统服务是处于在线状态还是离线状态。  l   由于资源转换到不正常操作，必须终止正常操作所产生的事件。  l   在应用程序正常运行过程中的错误散转情况。  在VISA中，定义了两种事件处理机制。一种是事件排队方式（Queuing），它主要分两步进行：首先允许特定事件的处理机制，然后在一定时刻点去查询事件是否已经发生（调用VISA中的viWaitOnEvent()函数，事件类型见VI\_EVENT\_类参数），在事件发生之后，必须关闭特定事件的处理机制，以防止资源溢出。例3.4所示为用事件排队方式处理VISA事件。  例3.4 排队方式处理VISA事件  另一种事件处理方式为回调方式（Callback），也即中断方式。它首先安装回调句柄，然后在程序中允许事件处理机制，并应编写相应的中断子程序。利用回调方式进行处理的事件类型主要有以下几种：  l   VI\_EVENT\_SERVICE\_REQ：该事件表示VXI器件服务请求  l   VI\_EVENT\_TRIG：该事件表示从VXI器件接收到硬件触发信号  l   VI\_EVENT\_VXI\_SIGP：该事件表示从VXI器件接收到总线信号或总线中断信号  例3.5、例3.6与例3.7分别是利用回调方式对各种类型事件进行处理的实例，其中  例3.5是对HP公司推出的VXI数字万用表模块进行服务请求事件的处理，而例3.6与  例3.7是对浙江大学数字所自主开发的VXI数字输入/输出模块进行触发事件与总线中断事件处理。事实上，这几种事件处理示例可直接推广应用到其它的消息基器件与寄存器基器件中。  例3.5 利用回调方式对VXI模块进行服务请求事件处理：  例3.5 回调方式进行服务请求事件处理  在这种模式中，主程序安装的事件句柄类型为VI\_EVENT\_SERVICE\_REQ，并须将服务请求使能。当服务请求事件发生时，安装的事件句柄被激活，系统进入中断子程序。中断子程序首先对事件类型进行判断，然后进行具体事件处理。在主程序退回时，必须关闭事件允许机制，并将安装的事件句柄进行卸载。  例3.6 利用回调方式对VXI模块进行触发事件处理：  例3.6 回调方式进行触发事件处理  在这种模式中，主程序安装的事件句柄类型为VI\_EVENT\_TRIG，并须将服务请求使能。本主程序利用设置仪器属性的方式进行触发总线的选择与有效，并利用软件进行同步触发信号线TTL0。当触发事件发生时，安装的事件句柄被激活，系统进入中断子程序。中断子程序首先对事件类型进行判断，然后进行具体事件处理。需要注意的是，在中断子程序中获取事件的句柄不再是仪器句柄vi，而是ViEvent类型的事件句柄ctx。在主程序退回时，必须将安装的事件句柄进行卸载。  例3.7 利用回调方式对VXI模块进行总线中断事件处理：  例3.7 回调方式进行总线中断事件处理  在这种模式中，主程序安装的事件句柄类型为VI\_EVENT\_VXI\_SIGP，并须将服务请求使能。本主程序直接对器件内部控制寄存器进行写操作，允许中断产生并强制中断产生。当触发事件发生时，安装的事件句柄被激活，系统进入中断子程序。中断子程序首先对事件类型进行判断，然后进行具体事件处理。同样，在中断子程序中获取事件的句柄不再是仪器句柄vi，而是ViEvent类型的事件句柄ctx。在主程序退回时，必须将安装的事件句柄进行卸载。  4、 资源操作集：包括了一系列VISA资源操作。而每个操作又分别包括以下各项：  l   名字（含形参名）：用于操作函数原型的简单描述。  l   目标：简要描述相应的操作功能。  l   参数表：列出了操作中所有参数的相关信息，包括参数名、输入/输出方向、参数数据类型及参数描述。操作参数表如表3.6所示。  参数名 输入/输出方向   数据类型    描述  表3.6资源操作参数表  l   返回状态值：VISA资源操作返回状态值类型是唯一的，即为ViStatus。操作返回状态值包括完成代码与错误代码两类。  l   描述：对每个函数的功能及其中关键参数的含义等作较详细的描述。  l   相关操作项：可参考的其它相关操作。  l   实现要求：操作实现时应满足的一些条件与备注要求。 |

续

|  |
| --- |
| 4.2.    viQueryRequest（vi，operationMode，command，param，len，response）  1） 目标：请求查询并返回响应值  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  operationMode   输入    ViUInt16    操作类型  command 输入    ViUInt32    命令列表  param   输入    ViAddr 参数值  len 输入/输出   ViUInt32    响应者数据长度  response    输出    ViAddr 命令响应  3） 返回状态值：  完成代码：  VI\_SUCCESS 查询完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_RSRC\_LOCKED    存取锁定模式不支持特定操作  VI\_ERROR\_CMD\_TMO    命令发送超时错误  VI\_ERROR\_RESP\_TMO   命令响应超时错误  VI\_ERROR\_BERR   总线错误  VI\_ERROR\_ABORT 传送时用户退出  VI\_ERROR\_RAW\_RD\_PROT\_VIOL读协议错误  VI\_ERROR\_RAW\_WR\_PROT\_VIOL写协议错误  VI\_ERROR\_OUTP\_PROT\_VIOL 输出协议错误  VI\_ERROR\_INP\_PROT\_VIOL 输入协议错误  VI\_ERROR\_NSUP\_COMMAND   命令未收到  VI\_ERROR\_RESP\_PENDING   前个响应正在登录  4） 描述：该操作可以向器件发送命令或接收上一个命令的响应。参数operationMode表明了操作类型。  5） 相关项：参见viSendCommand（）。  6） 实现要求：无。  4.3.    viGenProtError（vi，protError）  1） 目标：产生一个协议错误  2） 参数表：  参数名 输入/输出方向   数据类型    描述  vi 输入    ViSession   对话通道标识符（句柄）  protError   输入    ViUInt16    产生的协议错误  3） 返回状态值：  完成代码：  VI\_SUCCESS 协议错误产生完成  错误代码：  VI\_ERROR\_INV\_SESSION    vi不能标识正当对话通道  VI\_ERROR\_NSUP\_SERV\_OP   从机功能不支持  4） 描述：该操作产生的协议错误包括：清除任一协议错误状态、前一个响应正在登录引发的多个查询错误、命令未被识别、输入协议错误、输出协议错误、读协议错误与写协议错误等。  5） 相关项：无。  6） 实现要求：无。  n   CPU接口资源（VI\_RSRC\_CPU\_INTF）：  1. 资源概述：可实现对当地控制器的接口配置。  2. 资源属性表及属性描述：  属性名          描述  VI\_ATTR\_HW\_INTR\_LEVEL   CPU板中断级  VI\_ATTR\_IO\_BASE CPU I/O基地址  VI\_ATTR\_LOCAL\_BUS\_TMO   当地总线超时期间  VI\_ATTR\_MEM\_BASE    内存基地址  VI\_ATTR\_MSTR\_DMA\_CHNL   主机DMA通道  VI\_ATTR\_SLAVE\_DMA\_CHNL 从机DMA通道  VI\_ATTR\_SYSRESET\_RESET\_PC   SYSRESET是否会使当地控制器复位  VI\_ATTR\_LOCAL\_BYTE\_ORDER    接口控制器的字节顺序  3. CPU接口资源定义的事件如下所示：无  4. CPU接口资源定义的操作如下：无 |