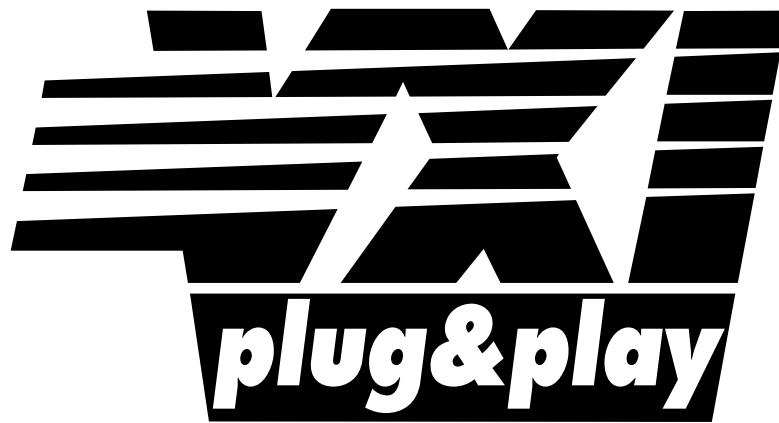


Systems Alliance

VPP-4.3.2: VISA Implementation Specification for Textual Languages

June 9, 2010

Revision 5.0



Systems Alliance

VPP-4.3.2 Revision History

This section is an overview of the revision history of the VPP-4.3.2 specification.

Revision 1.0, December 29, 1995

Original VISA document. Changes from VISA Transition Library include bindings for locking, asynchronous I/O, 32-bit register access, block moves, shared memory operations, and serial interface support.

Revision 1.1, January 22, 1997

Added new attributes, error codes, events, and formatted I/O modifiers.

Revision 2.0, December 5, 1997

Added error handling event, more formatted I/O operations, more serial attributes and extended searching capabilities. Changed ANSI C representation of attribute and event constants from ending in "L" to "UL" because they are all unsigned values.

Revision 2.0.1, December 4, 1998

Added new types to visatype.h for instrument drivers. Added new modes to give more robust functionality to viGpibControlREN. Updated information regarding contacting the Alliance.

Revision 2.2, November 19, 1999

Added new resource classes for GPIB (INTFC and SERVANT), VXI (BACKPLANE and SERVANT), and TCPIP (INSTR, SOCKET, and SERVANT).

Revision 3.0 Draft, January 14, 2003

Added new resource class for USB (INSTR). Removed definitions for the obsolete WIN framework (Windows 3.x), but this does not preclude a vendor implementation of VISA 3.0 on that framework.

Revision 3.0, January 15, 2004

Approved at IVI Board of Directors meeting.

Revision 4.0 Draft, May 16, 2006

Added new resource class for PXI (INSTR) to incorporate PXISA extensions. Added 64-bit extensions for register-based operations. Added support for WIN64 framework.

Revision 4.0, October 12, 2006

Approved at IVI Board of Directors meeting.

Revision 4.1, February 14, 2008

Updated the introduction to reflect the IVI Foundation organization changes. Replaced Notice with text used by IVI Foundation specifications.

Revision 4.1, April 14, 2008

Editorial change to update the IVI Foundation contact information in the Important Information section to remove obsolete address information and refer only to the IVI Foundation web site.

Revision 5.0, June 9, 2010

Added support for new TCPIP INSTR attributes regarding HiSLIP devices.

NOTICE

VPP-4.3.2: *VISA Implementation Specification for Textual Languages* is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at www.ivifoundation.org.

Warranty

The IVI Foundation and its member companies make no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The IVI Foundation and its member companies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Trademarks

Product and company names listed are trademarks or trade names of their respective companies.

No investigation has been made of common-law trademark rights in any work.

Table of Contents

Section 1	Introduction to the VXI <i>plug&play</i> Systems Alliance and the IVI Foundation.....	1-1
Section 2	Overview of VISA Implementation Specification.....	2-1
2.1	Objectives of This Specification.....	2-1
2.2	Audience for This Specification.....	2-1
2.3	Scope and Organization of This Specification.....	2-2
2.4	Application of This Specification.....	2-2
2.5	References.....	2-2
2.6	Definition of Terms and Acronyms.....	2-3
2.7	Conventions.....	2-6
Section 3	VISA Textual Language Bindings.....	3-1
3.1	Type Assignments.....	3-1
3.1.1	Type Assignments for WIN95 and WINNT Frameworks.....	3-5
3.1.2	Type Assignments for WIN64 Framework.....	3-5
3.2	Operation Prototypes.....	3-6
3.2.1	Operation Prototypes for WIN95 and WINNT Frameworks.....	3-10
3.2.2	Operation Prototypes for WIN64 Framework.....	3-14
3.3	Completion and Error Codes.....	3-15
3.4	Attribute Values.....	3-19
3.5	Event Type Values.....	3-24
3.6	Values and Ranges.....	3-25
3.7	Library Requirements.....	3-28
3.7.1	Library Requirements for WINNT and WIN64 Frameworks.....	3-28
3.8	Miscellaneous.....	3-31
Appendix A	Implementation Files.....	A-1
A.1	Contents of visatype.h File.....	A-1
A.2	Contents of visa.h File.....	A-4
A.3	Contents of visa32.bas File.....	A-15
A.4	Contents of visa32.def File.....	A-24
A.5	Contents of visa64.def File.....	A-26

Tables

Table 3.1.1.	Type Assignments for VISA and Instrument Drivers.....	3-1
Table 3.1.2.	Type Assignments for VISA Only.....	3-4
Table 3.2.1.	ANSI C Bindings for VISA Operations.....	3-6
Table 3.2.2.	Visual Basic Bindings for VISA Operations for the WIN95 and WINNT Frameworks.....	3-11
Table 3.3.1.	Completion and Error Codes.....	3-15
Table 3.4.1.	Attribute Values.....	3-19
Table 3.5.1.	Event Type Values.....	3-24
Table 3.6.1.	Values and Ranges.....	3-25
Table 3.7.1.	Procedure Definition Exports for the WINNT and WIN64 Frameworks.....	3-28
Table 3.8.1.	Bit Pattern for Attributes.....	3-33
Table 3.8.2.	Bit Pattern for Status Codes.....	3-33

Section 1 Introduction to the VXIplug&play Systems Alliance and the IVI Foundation

The VXIplug&play Systems Alliance was founded by members who shared a common commitment to end-user success with open, multivendor VXI systems. The alliance accomplished major improvements in ease of use by endorsing and implementing common standards and practices in both hardware and software, beyond the scope of the VXIbus specifications. The alliance used both formal and de facto standards to define complete system frameworks. These standard frameworks gave end-users "plug & play" interoperability at both the hardware and system software level.

The IVI Foundation is an organization whose members share a common commitment to test system developer success through open, powerful, instrument control technology. The IVI Foundation's primary purpose is to develop and promote specifications for programming test instruments that simplify interchangeability, provide better performance, and reduce the cost of program development and maintenance.

In 2002, the VXIplug&play Systems Alliance voted to become part of the IVI Foundation. In 2003, the VXIplug&play Systems Alliance formally merged into the IVI Foundation. The IVI Foundation has assumed control of the VXIplug&play specifications, and all ongoing work will be accomplished as part of the IVI Foundation.

All references to VXIplug&play Systems Alliance within this document, except contact information, were maintained to preserve the context of the original document.

Section 2 Overview of VISA Implementation Specification

This section introduces the VISA Implementation Specification for Textual Languages. This specification is a document authored by the *VXIplug&play* Systems Alliance. The technical work embodied in this document and the writing of this document was performed by the VISA Technical Working Group.

This section provides a complete overview of the VISA implementation specification, and gives readers general information that may be required to understand how to read, interpret, and implement individual aspects of this specification. This section is organized as follows:

- Objectives of this specification
- Audience for this specification
- Scope and organization of this specification
- Application of this specification
- References
- Definitions of terms and acronyms
- Conventions
- Communication

2.1 Objectives of This Specification

VISA gives VXI and GPIB software developers, particularly instrument driver developers, the functionality needed by instrument drivers in an interface-independent fashion for MXI, embedded VXI, GPIB-VXI, GPIB, and asynchronous serial controllers. *VXIplug&play* drivers written to the VISA specifications can execute on *VXIplug&play* system frameworks that have the VISA I/O library.

The VISA specification provides a common standard for the *VXIplug&play* System Alliance for developing multi-vendor software programs, including instrument drivers. This specification describes the VISA software model and the VISA Application Programming Interface (API).

The VISA Implementation Specification for Textual Languages addresses particular issues related to implementing source and binary level compatibility within specific frameworks, for the C and BASIC languages. Implementation issues for the G language are described in VPP-4.3.3: *VISA Implementation Specification for the G Language*.

2.2 Audience for This Specification

There are three audiences for this specification. The first audience is instrument driver developers—whether an instrument vendor, system integrator, or end user—who want to implement instrument driver software that is compliant with the *VXIplug&play* standards. The second audience is I/O vendors who want to implement VISA-compliant I/O software. The third audience is instrumentation end users and application programmers who want to implement applications that utilize instrument drivers compliant with this specification.

2.3 Scope and Organization of This Specification

This specification is organized in sections, with each section discussing a particular aspect of the VISA model.

Section 1 explains the VXI*plug&play* Systems Alliance and its relation to the IVI Foundation.

Section 2 provides an overview of this specification, including the objectives, scope and organization, application, references, definition of terms and acronyms, and conventions.

Section 3 provides the details of the VISA bindings to specific frameworks.

2.4 Application of This Specification

This specification is intended for use by developers of VXI*plug&play* instrument drivers and by developers of VISA I/O software. It is also useful as a reference for end users of VXI*plug&play* instrument drivers. This specification is intended to be used in conjunction with the VPP-3.x specifications, including the *Instrument Drivers Architecture and Design Specification* (VPP-3.1), the *Instrument Driver Functional Body Specification* (VPP-3.2), the *Instrument Interactive Developer Interface Specification* (VPP-3.3), and the *Instrument Driver Programmatic Developer Interface Specification* (VPP-3.4). These related specifications describe the implementation details for specific instrument drivers that are used with specific system frameworks. VXI*plug&play* instrument drivers developed in accordance with these specifications can be used in a wide variety of higher-level software environments, as described in the *System Frameworks Specification* (VPP-2).

2.5 References

The following documents contain information that you may find helpful as you read this document:

- ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*
- ANSI/IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols, and Common Commands*
- ANSI/IEEE Standard 1014-1987, *IEEE Standard for a Versatile Backplane Bus: VMEbus*
- *ANSI/IEEE Standard 1174-2000, Standard Serial Interface for Programmable Instrumentation*

- VPP-1, *VXIplug&play Charter Document*
- VPP-2, *System Frameworks Specification*
- VPP-3.1, *Instrument Drivers Architecture and Design Specification*
- VPP-3.2, *Instrument Driver Functional Body Specification*
- VPP-3.3, *Instrument Driver Interactive Developer Interface Specification*
- VPP-3.4, *Instrument Driver Programmatic Developer Interface Specification*
- VPP-4.3, *The VISA Library*
- VPP-4.3.3, *VISA Implementation Specification for the G Language*
- VPP-6, *Installation and Packaging Specification*
- VPP-7, *Soft Front Panel Specification*
- VPP-9, *Instrument Vendor Abbreviations*
- VXI-1, *VXIbus System Specification*, Revision 1.4, VXIbus Consortium

2.6 Definition of Terms and Acronyms

The following are some commonly used terms within this document

Address	A string (or other language construct) that uniquely locates and identifies a resource. VISA defines an ASCII-based grammar that associates strings with particular physical devices or interfaces and VISA resources.
ADE	Application Development Environment
API	Application Programmers Interface. The direct interface that an end user sees when creating an application. The VISA API consists of the sum of all of the operations, attributes, and events of each of the VISA Resource Classes.
Attribute	A value within a resource that reflects a characteristic of the operational state of a resource.
Bus Error	An error that signals failed access to an address. Bus errors occur with low-level accesses to memory and usually involve hardware with bus mapping capabilities. For example, non-existent memory, a non-existent register, or an incorrect device access can cause a bus error.
Commander	A device that has the ability to control another device. This term can also denote the unique device that has sole control over another device (as with the VXI Commander/Servant hierarchy).
Communication Channel	The same as <i>Session</i> . A communication path between a software element and a resource. Every communication channel in VISA is unique.

Controller	A device that can control another device(s) or is in the process of performing an operation on another device.
Device	An entity that receives commands from a controller. A device can be an instrument, a computer (acting in a non-controller role), or a peripheral (such as a plotter or printer). In VISA, the concept of a device is generally the logical association of several VISA resources.
Instrument	A device that accepts some form of stimulus to perform a designated task, test, or measurement function. Two common forms of stimuli are message passing and register reads and writes. Other forms include triggering or varying forms of asynchronous control.
Interface	A generic term that applies to the connection between devices and controllers. It includes the communication media and the device/controller hardware necessary for cross-communication.
Instrument Driver	Library of functions for controlling a specific instrument
Mapping	An operation that returns a reference to a specified section of an address space and makes the specified range of addresses accessible to the requester. This function is independent of memory allocation.
Operation	An action defined by a resource that can be performed on a resource.
Process	An operating system component that shares a system's resources. A multi-process system is a computer system that allows multiple programs to execute simultaneously, each in a separate process environment. A single-process system is a computer system that allows only a single program to execute at a given point in time.
Register	An address location that either contains a value that is a function of the state of hardware or can be written into to cause hardware to perform a particular action or to enter a particular state. In other words, an address location that controls and/or monitors hardware.
Resource Class	The definition for how to create a particular resource. In general, this is synonymous with the connotation of the word <i>class</i> in object-oriented architectures. For VISA Instrument Control Resource Classes, this refers to the definition for how to create a resource that controls a particular capability of a device.
Resource or Resource Instance	In general, this term is synonymous with the connotation of the word <i>object</i> in object-oriented architectures. For VISA, <i>resource</i> more specifically refers to a particular implementation (or <i>instance</i> in object-oriented terms) of a Resource Class. In VISA, every defined software module is a resource.
Session	The same as <i>Communication Channel</i> . A communication path between a software element and a resource. Every communication channel in VISA is unique.

SRQ	IEEE 488 Service Request. This is an asynchronous request from a remote GPIB device that requires service. A service request is essentially an interrupt from a remote device. For GPIB, this amounts to asserting the SRQ line on the GPIB. For VXI, this amounts to sending the Request for Service True event (REQT).
Status Byte	A byte of information returned from a remote device that shows the current state and status of the device. If the device follows IEEE 488 conventions, bit 6 of the status byte indicates if the device is currently requesting service.
Template Function	Instrument driver subsystem function common to the majority of <i>VXIplug&play</i> instrument drivers
Top-level Example	A high-level test-oriented instrument driver function. It is typically developed from the instrument driver subsystem functions.
Virtual Instrument	A name given to the grouping of software modules (in this case, VISA resources with any associated or required hardware) to give the functionality of a traditional stand-alone instrument. Within VISA, a virtual instrument is the logical grouping of any of the VISA resources. The VISA Instrument Control Resources Organizer serves as a means to group any number of any type of VISA Instrument Control Resources within a VISA system.
VISA	Virtual Instrument Software Architecture. This is the general name given to this document and its associated architecture. The architecture consists of two main VISA components: the VISA Resource Manager and the VISA Instrument Control Resources.
VISA Instrument Control Resources	This is the name given to the part of VISA that defines all of the device-specific resource classes. VISA Instrument Control Resources encompass all defined device and interface capabilities for direct, low-level instrument control.
VISA Resource Manager	This is the name given to the part of VISA that manages resources. This management includes support for opening, closing, and finding resources; setting attributes, retrieving attributes, and generating events on resources; and so on.
VISA Resource Template	This is the name given to the part of VISA that defines the basic constraints and interface definition for the creation and use of a VISA resource. All VISA resources must derive their interface from the definition of the VISA Resource Template.

2.7 Conventions

Throughout this specification you will see the following headings on certain paragraphs. These headings instill special meaning on these paragraphs.

Rules must be followed to ensure compatibility with the System Framework. A rule is characterized by the use of the words **SHALL** and **SHALL NOT** in bold upper case characters. These words are not used in this manner for any other purpose other than stating rules.

Recommendations consist of advice to implementors which will affect the usability of the final device. They are included in this standard to draw attention to particular characteristics which the authors believe to be important to end user success.

Permissions are included to *authorize* specific implementations or uses of system components. A permission is characterized by the use of the word **MAY** in bold upper case characters. These permissions are granted to ensure specific System Framework components are well defined and can be tested for compatibility and interoperability.

Observations spell out implications of rules and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain rules, so that the reader understands why the rule must be followed.

A note on the text of the specification: Any text which appears without heading should be considered as description of the standard and how the architecture was intended to operate. The purpose of this text is to give the reader a deeper understanding of the intentions of the specification including the underlying model and specific required features. As such, the implementor of this standard should take great care to ensure that a particular implementation does not conflict with the text of the standard.

Section 3 VISA Textual Language Bindings

3.1 Type Assignments

Tables 3.1.1 and 3.1.2 give the type assignments for ANSI C and Visual Basic for each type defined in VPP-4.3. Although ANSI C types can be defined in a header file, Visual Basic types cannot. Table 3.1.1 lists those types that are both used and exported by direct users of VISA (such as instrument drivers). Table 3.1.2 lists types that may be used but not exported by such users. For example, end-users would see the types specified in Table 3.1.1 exported by the instrument driver; however, they would not see the types specified in Table 3.1.2.

Table 3.1.1. Type Assignments for VISA and Instrument Drivers

VISA Data Type	ANSI C Binding	Visual Basic Binding	Description
ViUInt32	unsigned long	Long	A 32-bit unsigned integer.
ViPUInt32	ViUInt32 *	N/A	The location of a 32-bit unsigned integer.
ViAUInt32	ViUInt32[]	N/A	An array of 32-bit unsigned integers.
ViInt32	signed long	Long	A 32-bit signed integer.
ViPInt32	ViInt32 *	N/A	The location of a 32-bit signed integer.
ViAInt32	ViInt32[]	N/A	An array of 32-bit signed integers.
ViUInt64	Unsigned int64 or u_int64_t	N/A	A 64-bit unsigned integer. The exact type definition depends on the compiler.
ViPUInt64	ViUInt64 *	N/A	The location of a 64-bit unsigned integer.
ViAUInt64	ViUInt64[]	N/A	An array of 64-bit unsigned integers.
ViInt64	signed int64 or int64_t	N/A	A 64-bit signed integer. The exact type definition depends on the compiler.
ViPInt64	ViInt64 *	N/A	The location of a 64-bit signed integer.
ViAInt64	ViInt64[]	N/A	An array of 64-bit signed integers.
ViUInt16	unsigned short	Integer	A 16-bit unsigned integer.
ViPUInt16	ViUInt16 *	N/A	The location of a 16-bit unsigned integer.
ViAUInt16	ViUInt16[]	N/A	An array of 16-bit unsigned integers.
ViInt16	signed short	Integer	A 16-bit signed integer.
ViPInt16	ViInt16 *	N/A	The location of a 16-bit signed integer.
ViAInt16	ViInt16[]	N/A	An array of 16-bit signed integers.
ViUInt8	unsigned char	Integer/ Byte	An 8-bit unsigned integer.
ViPUInt8	ViUInt8 *	N/A	The location of an 8-bit unsigned integer.
ViAUInt8	ViUInt8[]	N/A	An array of 8-bit unsigned integers.

(continues)

Table 3.1.1. Type Assignments for VISA and Instrument Drivers (Continued)

VISA Data Type	ANSI C Binding	Visual Basic Binding	Description
ViInt8	signed char	Integer/ Byte	An 8-bit signed integer.
ViPInt8	ViInt8 *	N/A	The location of an 8-bit signed integer.
ViAInt8	ViInt8[]	N/A	An array of 8-bit signed integers.
ViAddr	void *	Long	A type that references another data type, in cases where the other data type may vary depending on a particular context.
ViPAddr	ViAddr *	N/A	The location of a ViAddr.
ViAAddr	ViAddr[]	N/A	An array of type ViAddr.
ViChar	char	Integer/ Byte	An 8-bit integer representing an ASCII character.
ViPChar	ViChar *	N/A	The location of a ViChar.
ViAChar	ViChar[]	N/A	An array of type ViChar.
ViByte	unsigned char	Integer/ Byte	An 8-bit unsigned integer representing an extended ASCII character.
ViPByte	ViByte *	N/A	The location of a ViByte.
ViAByte	ViByte[]	N/A	An array of type ViByte.
ViBoolean	ViUInt16	Integer	A type for which there are two complementary values: VI_TRUE and VI_FALSE.
ViPBoolean	ViBoolean *	N/A	The location of a ViBoolean.
ViABoolean	ViBoolean[]	N/A	An array of type ViBoolean.
ViReal32	float	Single	A 32-bit single-precision value.
ViPReal32	ViReal32 *	N/A	The location of a 32-bit single-precision value.
ViAReal32	ViReal32[]	N/A	An array of 32-bit single-precision values.
ViReal64	double	Double	A 64-bit double-precision value.
ViPReal64	ViReal64 *	N/A	The location of a 64-bit double-precision value.
ViAReal64	ViReal64[]	N/A	An array of 64-bit double-precision values.
ViBuf	ViPByte	String	The location of a block of data.
ViPBuf	ViPByte	String	The location to store a block of data.
ViABuf	ViBuf[]	N/A	An array of type ViBuf.
ViString	ViPChar	String	The location of a NULL-terminated ASCII string.
ViPString	ViPChar	String	The location to store a NULL-terminated ASCII string.
ViAString	ViString[]	N/A	An array of type ViString.

(continues)

Table 3.1.1. Type Assignments for VISA and Instrument Drivers (Continued)

VISA Data Type	ANSI C Binding	Visual Basic Binding	Description
ViRsrc	ViString	String	A ViString type that is further restricted to adhere to the addressing grammar for resources as presented in Section 3 of VPP-4.3.
ViPRsrc	ViString	String	The location to store a ViRsrc.
ViARsrc	ViRsrc[]	N/A	An array of type ViRsrc.
ViStatus	ViInt32	Long	A defined type that contains values corresponding to VISA-defined Completion and Error termination codes.
ViPStatus	ViStatus *	N/A	The location of a ViStatus.
ViAStatus	ViStatus[]	N/A	An array of type ViStatus.
ViVersion	ViUInt32	Long	A defined type that contains a reference to all information necessary for the architect to represent the current version of a resource.
ViPVersion	ViVersion *	N/A	The location of a ViVersion.
ViAVersion	ViVersion[]	N/A	An array of type ViVersion.
ViObject	ViUInt32	Long	The most fundamental VISA data type. It contains attributes and can be closed when no longer needed.
ViPObject	ViObject *	N/A	The location of a ViObject.
ViAObject	ViObject[]	N/A	An array of type ViObject.
ViSession	ViObject	Long	A defined type that contains a reference to all information necessary for the architect to manage a communication channel with a resource.
ViPSession	ViSession *	N/A	The location of a ViSession.
ViASession	ViSession[]	N/A	An array of type ViSession.
ViAttr	ViUInt32	Long	A type that uniquely identifies an attribute.
ViConstString	const ViChar *	String	A ViString type that is guaranteed to not be modified by any driver.

OBSERVATION 3.1.1

Table 3.1.1 lists each fundamental data type, a second type that is a reference to the fundamental data type, and a third type that indicates an array of the fundamental data type. For example, the entry ViUInt32, at the beginning of Table 3.1.1, is a fundamental data type. Fundamental data types are used for variable declarations and input parameters. ViPUInt32 is a reference to a ViUInt32, and is used for output parameters of type ViUInt32. ViAUInt32 is used for both input and output parameters of arrays of type ViUInt32.

OBSERVATION 3.1.2

In the case of Visual Basic, input parameters are passed by value (ByVal). Output parameters are not passed by value except for string types. For arrays, the first element of the array should be passed by reference. For example, to pass an array x, use x(0).

Table 3.1.2. Type Assignments for VISA Only

VISA Data Type	ANSI C Binding	Visual Basic Binding	Description
ViAccessMode	ViUInt32	Long	A defined type that specifies the different mechanisms that control access to a resource.
ViPAccessMode	ViAccessMode *	N/A	The location of a ViAccessMode.
ViBusAddress	ViUInt32 or ViUInt64	Long or N/A	A type that represents the system-dependent physical address. This varies on 32-bit and 64-bit systems.
ViBusAddress64	ViUInt64	N/A	A type that represents a physical address that is always 64 bits, even on 32-bit systems.
ViPBusAddress	ViBusAddress *	N/A	The location of a ViBusAddress.
ViPBusAddress64	ViBusAddress64 *	N/A	The location of a ViBusAddress64.
ViBusSize	ViUInt32 or ViUInt64	Long or N/A	A type that represents the system dependent physical address size. This varies on 32-bit and 64-bit systems.
ViAttrState	ViUInt32 or ViUInt64	Long or N/A	A value unique to the individual type of an attribute. This varies on 32-bit and 64-bit systems.
ViPAttrState	void *	Any	The location of a ViAttrState.
ViVAlList	va_list	Any	The location of a list of a variable number of parameters of differing types.
ViEventType	ViUInt32	Long	A defined type that uniquely identifies the type of an event.
ViPEventType	ViEventType *	N/A	The location of a ViEventType.
ViAEventType	ViEventType *	N/A	An array of type ViEventType.
ViPAttr	ViAttr *	N/A	The location of a ViAttr.
ViAAttr	ViAttr *	N/A	An array of type ViAttr.
ViEventFilter	ViUInt32	Long	A defined type that specifies filtering masks or other information unique to an event.
ViFindList	ViObject	Long	A defined type that contains a reference to all resources found during a search operation.
ViPFindList	ViFindList *	N/A	The location of a ViFindList.
ViEvent	ViObject	Long	A defined type that encapsulates the information necessary to process an event.
ViPEvent	ViEvent *	N/A	The location of a ViEvent.
ViKeyId	ViString	String	A defined type that contains a reference to all information necessary for the architect to manage the association of a thread or process and session with a lock on a resource.

(continues)

Table 3.1.2. Type Assignments for VISA Only (Continued)

VISA Data Type	ANSI C Binding	Visual Basic Binding	Description
ViPKeyId	ViPString	String	The location of a ViKeyId.
ViJobId	ViUInt32	Long	A defined type that contains a reference to all information necessary for the architect to encapsulate the information necessary for a posted operation request.
ViPJobId	ViJobId *	N/A	The location of a ViJobId.
ViHndlr	ViStatus (*) (ViSession, ViEventType, ViEvent, ViAddr)	N/A	A value representing an entry point to an operation for use as a callback.

OBSERVATION 3.1.3

The pointer type `ViHndlr` is a code pointer rather than a data pointer. Therefore, it must be treated differently in some frameworks.

RULE 3.1.1

All types in Tables 3.1.1 and 3.1.2 **SHALL** be defined to the specified bindings.

RULE 3.1.2

All ANSI C definitions in Table 3.1.1 **SHALL** be present within the `visatype.h` file.

RULE 3.1.3

All ANSI C definitions in Table 3.1.2 **SHALL** be present within the `visa.h` file.

3.1.1 Type Assignments for WINNT Framework**RULE 3.1.4**

Unless otherwise stated, all pointers in Tables 3.1.1 and 3.1.2 **SHALL** be treated as flat 32-bit data pointers when interfacing to the WINNT Framework DLL.

RULE 3.1.5

The pointer type `ViHndlr` **SHALL** be treated as a `_stdcall` pointer when interfacing to the WINNT Framework DLL.

3.1.2 Type Assignments for WIN64 Framework

NOTE: The definition of the WIN64 framework is currently in progress. Version 4.0 of the VISA family of specifications (VPP 4.3) refer to the WIN64 framework being defined in VPP 2 (Frameworks) and VPP 6 (Installation). When the definition of the WIN64 framework in VPP 2 and VPP 6 is complete, it will apply to the VISA 4.0 specifications and these “in progress” notes will be removed as an editorial change.

RULE 3.1.6

Unless otherwise stated, all pointers in Tables 3.1.1 and 3.1.2 **SHALL** be treated as flat 64-bit data pointers when interfacing to the WIN64 Framework DLL.

RULE 3.1.7

The pointer type `ViHndlr` **SHALL** be treated as a `fastcall` pointer when interfacing to the WIN64 Framework DLL.

3.2 Operation Prototypes

The following sections specify the operation prototypes for ANSI C and Visual Basic. Table 3.2.1 gives the function prototypes for the ANSI C bindings for each function and operation in VPP-4.3.

Table 3.2.1. ANSI C Bindings for VISA Operations

ViStatus viGetDefaultRM	(ViPSession sesn);
ViStatus viOpenDefaultRM	(ViPSession sesn);
ViStatus viFindRsrc	(ViSession sesn, ViString expr, ViPFindList findList, ViPUInt32 retCnt, ViChar _VI_FAR desc[]);
ViStatus viFindNext	(ViFindList findList, ViChar _VI_FAR desc[]);
ViStatus viParseRsrc	(ViSession sesn, ViRsrc rsrcName, ViPUInt16 intfType, ViPUInt16 intfNum);
ViStatus viParseRsrcEx	(ViSession sesn, ViRsrc rsrcName, ViPUInt16 intfType, ViPUInt16 intfNum, ViChar VI_FAR rsrcClass[], ViChar VI_FAR expandedUnaliasedName[], ViChar VI_FAR aliasIfExists[]);
ViStatus viOpen	(ViSession sesn, ViRsrc name, ViAccessMode mode, ViUInt32 timeout, ViPSession vi);
ViStatus viClose	(ViObject vi);
ViStatus viGetAttribute	(ViObject vi, ViAttr attrName, void _VI_PTR attrValue);
ViStatus viSetAttribute	(ViObject vi, ViAttr attrName, ViAttrState attrValue);
ViStatus viStatusDesc	(ViObject vi, ViStatus status, ViChar _VI_FAR desc[]);
ViStatus viTerminate	(ViObject vi, ViUInt16 degree, ViJobId jobId);
ViStatus viLock	(ViSession vi, ViAccessMode lockType, ViUInt32 timeout, ViKeyId requestedKey, ViChar _VI_FAR accessKey[]);
ViStatus viUnlock	(ViSession vi);
ViStatus viEnableEvent	(ViSession vi, ViEventType eventType, ViUInt16 mechanism, ViEventFilter context);
ViStatus viDisableEvent	(ViSession vi, ViEventType eventType, ViUInt16 mechanism);
ViStatus viDiscardEvents	(ViSession vi, ViEventType eventType, ViUInt16 mechanism);
ViStatus viWaitOnEvent	(ViSession vi, ViEventType inEventType, ViUInt32 timeout, ViPEventType outEventType, ViPEvent outContext);
ViStatus viInstallHandler	(ViSession vi, ViEventType eventType, ViHndlr handler, ViAddr userHandle);
ViStatus viUninstallHandler	(ViSession vi, ViEventType eventType, ViHndlr handler, ViAddr userHandle);
ViStatus viRead	(ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus viReadAsync	(ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPJobId jobId);
ViStatus viReadToFile	(ViSession vi, ViConstString filename, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus viWrite	(ViSession vi, ViBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);

(continues)

Table 3.2.1. ANSI C Bindings for VISA Operations (Continued)

ViStatus viWriteAsync	(ViSession vi, ViBuf buf, ViUInt32 cnt, ViPJobId jobId);
ViStatus viWriteFromFile	(ViSession vi, ViConstString filename, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus viAssertTrigger	(ViSession vi, ViUInt16 protocol);
ViStatus viReadSTB	(ViSession vi, ViPUInt16 status);
ViStatus viClear	(ViSession vi);
ViStatus viSetBuf	(ViSession vi, ViUInt16 mask, ViUInt32 size);
ViStatus viFlush	(ViSession vi, ViUInt16 mask);
ViStatus viBufWrite	(ViSession vi, ViBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus viBufRead	(ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus viPrintf	(ViSession vi, ViString writeFmt, ...);
ViStatus viVPrintf	(ViSession vi, ViString writeFmt, ViVAList params);
ViStatus viSPrintf	(ViSession vi, ViPBuf buf, ViString writeFmt, ...);
ViStatus viVSPrintf	(ViSession vi, ViPBuf buf, ViString writeFmt, ViVAList params);
ViStatus viScanf	(ViSession vi, ViString readFmt, ...);
ViStatus viVScanf	(ViSession vi, ViString readFmt, ViVAList params);
ViStatus viSScanf	(ViSession vi, ViBuf buf, ViString readFmt, ...);
ViStatus viVSScanf	(ViSession vi, ViBuf buf, ViString readFmt, ViVAList params);
ViStatus viQueryf	(ViSession vi, ViString writeFmt, ViString readFmt, ...);
ViStatus viVQueryf	(ViSession vi, ViString writeFmt, ViString readFmt, ViVAList params);
ViStatus viIn8	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViPUInt8 val8);
ViStatus viOut8	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViUInt8 val8);
ViStatus viIn16	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViPUInt16 val16);
ViStatus viOut16	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViUInt16 val16);
ViStatus viIn32	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViPUInt32 val32);
ViStatus viOut32	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViUInt32 val32);
ViStatus viIn64	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViPUInt64 val64);
ViStatus viOut64	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViUInt64 val64);
ViStatus viIn8Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViPUInt8 val8);
ViStatus viOut8Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViUInt8 val8);
ViStatus viIn16Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViPUInt16 val16);
ViStatus viOut16Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViUInt16 val16);
ViStatus viIn32Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViPUInt32 val32);
ViStatus viOut32Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViUInt32 val32);
ViStatus viIn64Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViPUInt64 val64);

(continues)

Table 3.2.1. ANSI C Bindings for VISA Operations (Continued)

ViStatus viOut64Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViUInt64 val64);
ViStatus viMoveIn8	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt8 buf8);
ViStatus viMoveOut8	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt8 buf8);
ViStatus viMoveIn16	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt16 buf16);
ViStatus viMoveOut16	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt16 buf16);
ViStatus viMoveIn32	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt32 buf32);
ViStatus viMoveOut32	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt32 buf32);
ViStatus viMoveIn64	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt64 buf64);
ViStatus viMoveOut64	(ViSession vi, ViUInt16 space, ViBusAddress offset, ViBusSize length, ViAUInt64 buf64);
ViStatus viMoveIn8Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt8 buf8);
ViStatus viMoveOut8Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt8 buf8);
ViStatus viMoveIn16Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt16 buf16);
ViStatus viMoveOut16Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt16 buf16);
ViStatus viMoveIn32Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt32 buf32);
ViStatus viMoveOut32Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt32 buf32);
ViStatus viMoveIn64Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt64 buf64);
ViStatus viMoveOut64Ex	(ViSession vi, ViUInt16 space, ViBusAddress64 offset, ViBusSize length, ViAUInt64 buf64);
ViStatus viMove	(ViSession vi, ViUInt16 srcSpace, ViBusAddress srcOffset, ViUInt16 srcWidth, ViUInt16 destSpace, ViBusAddress destOffset, ViUInt16 destWidth, ViBusSize srcLength);
ViStatus viMoveAsync	(ViSession vi, ViUInt16 srcSpace, ViBusAddress srcOffset, ViUInt16 srcWidth, ViUInt16 destSpace, ViBusAddress destOffset, ViUInt16 destWidth, ViBusSize srcLength, ViPJobId jobId);
ViStatus viMoveEx	(ViSession vi, ViUInt16 srcSpace, ViBusAddress64 srcOffset, ViUInt16 srcWidth, ViUInt16 destSpace, ViBusAddress64 destOffset, ViUInt16 destWidth, ViBusSize srcLength);
ViStatus viMoveAsyncEx	(ViSession vi, ViUInt16 srcSpace, ViBusAddress64 srcOffset, ViUInt16 srcWidth, ViUInt16 destSpace, ViBusAddress64 destOffset, ViUInt16 destWidth, ViBusSize srcLength, ViPJobId jobId);
ViStatus viMapAddress	(ViSession vi, ViUInt16 mapSpace, ViBusAddress mapOffset, ViBusSize mapSize, ViBoolean access, ViAddr suggested, ViPAddr address);
ViStatus viUnmapAddress	(ViSession vi);

(continues)

Table 3.2.1. ANSI C Bindings for VISA Operations (Continued)

ViStatus viMapAddressEx	(ViSession vi, ViUInt16 mapSpace, ViBusAddress64 mapOffset, ViBusSize mapSize, ViBoolean access, ViAddr suggested, ViPAddr address);
void viPeek8	(ViSession vi, ViAddr address, ViPUInt8 val8);
void viPoke8	(ViSession vi, ViAddr address, ViUInt8 val8);
void viPeek16	(ViSession vi, ViAddr address, ViPUInt16 val16);
void viPoke16	(ViSession vi, ViAddr address, ViUInt16 val16);
void viPeek32	(ViSession vi, ViAddr address, ViPUInt32 val32);
void viPoke32	(ViSession vi, ViAddr address, ViUInt32 val32);
void viPeek64	(ViSession vi, ViAddr address, ViPUInt64 val64);
void viPoke64	(ViSession vi, ViAddr address, ViUInt64 val64);
ViStatus viMemAlloc	(ViSession vi, ViBusSize size, ViPBusAddress offset);
ViStatus viMemFree	(ViSession vi, ViBusAddress offset);
ViStatus viMemAllocEx	(ViSession vi, ViBusSize size, ViPBusAddress64 offset);
ViStatus viMemFreeEx	(ViSession vi, ViBusAddress64 offset);
ViStatus viGpibControlREN	(ViSession vi, ViUInt16 mode);
ViStatus viGpibControlATN	(ViSession vi, ViUInt16 mode);
ViStatus viGpibSendIFC	(ViSession vi);
ViStatus viGpibCommand	(ViSession vi, ViBuf cmd, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus viGpibPassControl	(ViSession vi, ViUInt16 primAddr, ViUInt16 secAddr);
ViStatus viVxiCommandQuery	(ViSession vi, ViUInt16 mode, ViUInt32 cmd, ViPUInt32 response);
ViStatus viAssertUtilSignal	(ViSession vi, ViUInt16 line);
ViStatus viAssertIntrSignal	(ViSession vi, ViInt16 mode, ViUInt32 statusID);
ViStatus viMapTrigger	(ViSession vi, ViInt16 trigSrc, ViInt16 trigDest, ViUInt16 mode);
ViStatus viUnmapTrigger	(ViSession vi, ViInt16 trigSrc, ViInt16 trigDest);
ViStatus viUsbControlOut	(ViSession vi, ViInt16 bmRequestType, ViInt16 bRequest, ViUInt16 wValue, ViUInt16 wIndex, ViUInt16 wLength, ViBuf buf);
ViStatus viUsbControlIn	(ViSession vi, ViInt16 bmRequestType, ViInt16 bRequest, ViUInt16 wValue, ViUInt16 wIndex, ViUInt16 wLength, ViPBuf buf, ViPUInt16 retCnt);

RULE 3.2.1

All functions and operations specified in Table 3.2.1 **SHALL** be implemented as specified.

RULE 3.2.2

The ANSI C definitions in Table 3.2.1 **SHALL** be present within the `visa.h` file.

OBSERVATION 3.2.1

The operations `viPrintf()`, `viScanf()`, and `viQueryf()` take a variable number of arguments, which requires a different calling convention in some frameworks.

3.2.1 Operation Prototypes for WINNT Framework

RULE 3.2.3

Unless otherwise stated, all functions and operations specified in Table 3.2.1 **SHALL** be treated as `_stdcall` when interfacing to the WINNT Framework DLL.

RULE 3.2.4

The operations `viPrintf()`, `viScanf()`, and `viQueryf()` **SHALL** be treated as `_cdecl` when interfacing to the WINNT Framework DLL.

RULE 3.2.5

All pointers in Table 3.2.1 **SHALL** be treated as flat 32-bit pointers when interfacing to the WINNT Framework DLL.

Table 3.2.2 gives the function prototypes for the Visual Basic bindings for each operation in VPP-4.3 for the WINNT framework.

Table 3.2.2. Visual Basic Bindings for VISA Operations for the WINNT Framework

```

Declare Function viGetDefaultRM Lib "VISA32.DLL" Alias "#128" (sesn As Long)
    As Long
Declare Function viOpenDefaultRM Lib "VISA32.DLL" Alias "#141" (sesn As Long)
    As Long
Declare Function viFindRsrc Lib "VISA32.DLL" Alias "#129" (ByVal sesn As Long,
    ByVal expr As String, vi As Long, retCount As Long, ByVal
    desc As String) As Long
Declare Function viFindNext Lib "VISA32.DLL" Alias "#130" (ByVal vi As Long,
    ByVal desc As String) As Long
Declare Function viParseRsrc Lib "VISA32.DLL" Alias "#146" (ByVal sesn As
    Long, ByVal desc As String, intfType As Integer, intfNum
    As Integer) As Long
Declare Function viParseRsrcEx Lib "VISA32.DLL" Alias "#147" (ByVal sesn As
    Long, ByVal desc As String, intfType As Integer, intfNum
    As Integer, ByVal rsrcClass As String, ByVal
    expandedUnaliasedName As String, ByVal aliasIfExists As
    String) As Long
Declare Function viOpen Lib "VISA32.DLL" Alias "#131" (ByVal sesn As Long,
    ByVal desc As String, ByVal mode As Long, ByVal timeout
    As Long, vi As Long) As Long
Declare Function viClose Lib "VISA32.DLL" Alias "#132" (ByVal vi As Long) As
    Long
Declare Function viGetAttribute Lib "VISA32.DLL" Alias "#133" (ByVal vi As
    Long, ByVal attrName As Long, attrValue As Any) As Long
Declare Function viSetAttribute Lib "VISA32.DLL" Alias "#134" (ByVal vi As
    Long, ByVal attrName As Long, ByVal attrValue As Long) As
    Long
Declare Function viStatusDesc Lib "VISA32.DLL" Alias "#142" (ByVal vi As Long,
    ByVal status As Long, ByVal desc As String) As Long
Declare Function viLock Lib "VISA32.DLL" Alias "#144" (ByVal vi As Long, ByVal
    lockType As Long, ByVal timeout As Long, ByVal
    requestedKey As String, ByVal accessKey As String) As
    Long
Declare Function viUnlock Lib "VISA32.DLL" Alias "#145" (ByVal vi As Long) As
    Long
Declare Function viEnableEvent Lib "VISA32.DLL" Alias "#135" (ByVal vi As
    Long, ByVal eventType As Long, ByVal mechanism As
    Integer, ByVal context As Long) As Long
Declare Function viDisableEvent Lib "VISA32.DLL" Alias "#136" (ByVal vi As
    Long, ByVal eventType As Long, ByVal mechanism As
    Integer) As Long
Declare Function viDiscardEvents Lib "VISA32.DLL" Alias "#137" (ByVal vi As
    Long, ByVal eventType As Long, ByVal mechanism As
    Integer) As Long
Declare Function viWaitOnEvent Lib "VISA32.DLL" Alias "#138" (ByVal vi As
    Long, ByVal inEventType As Long, ByVal timeout As Long,
    outEventType As Long, outEventContext As Long) As Long
Declare Function viRead Lib "VISA32.DLL" Alias "#256" (ByVal vi As Long, ByVal
    Buffer As String, ByVal count As Long, retCount As Long)
    As Long

```

(continues)

Table 3.2.2. Visual Basic Bindings for VISA Operations for the WINNT Framework (Continued)

Declare Function viReadToFile Lib "VISA32.DLL" Alias "#219" (ByVal vi As Long, ByVal filename As String, ByVal count As Long, retCount As Long) As Long
Declare Function viWrite Lib "VISA32.DLL" Alias "#257" (ByVal vi As Long, ByVal Buffer As String, ByVal count As Long, retCount As Long) As Long
Declare Function viWriteFromFile Lib "VISA32.DLL" Alias "#218" (ByVal vi As Long, ByVal filename As String, ByVal count As Long, retCount As Long) As Long
Declare Function viAssertTrigger Lib "VISA32.DLL" Alias "#258" (ByVal vi As Long, ByVal protocol As Integer) As Long
Declare Function viReadSTB Lib "VISA32.DLL" Alias "#259" (ByVal vi As Long, status As Integer) As Long
Declare Function viClear Lib "VISA32.DLL" Alias "#260" (ByVal vi As Long) As Long
Declare Function viBufWrite Lib "VISA32.DLL" Alias "#202" (ByVal vi As Long, ByVal Buffer As String, ByVal count As Long, retCount As Long) As Long
Declare Function viBufRead Lib "VISA32.DLL" Alias "#203" (ByVal vi As Long, ByVal Buffer As String, ByVal count As Long, retCount As Long) As Long
Declare Function viSetBuf Lib "VISA32.DLL" Alias "#267" (ByVal vi As Long, ByVal mask As Integer, ByVal bufSize As Long) As Long
Declare Function viFlush Lib "VISA32.DLL" Alias "#268" (ByVal vi As Long, ByVal mask As Integer) As Long
Declare Function viVPrintf Lib "VISA32.DLL" Alias "#270" (ByVal vi As Long, ByVal writeFmt As String, params As Any) As Long
Declare Function viVSPrintf Lib "VISA32.DLL" Alias "#205" (ByVal vi As Long, ByVal Buffer As String, ByVal writeFmt As String, params As Any) As Long
Declare Function viVScanf Lib "VISA32.DLL" Alias "#272" (ByVal vi As Long, ByVal readFmt As String, params As Any) As Long
Declare Function viVSScanf Lib "VISA32.DLL" Alias "#207" (ByVal vi As Long, ByVal Buffer As String, ByVal readFmt As String, params As Any) As Long
Declare Function viVQueryf Lib "VISA32.DLL" Alias "#280" (ByVal vi As Long, ByVal writeFmt As String, ByVal readFmt As String, params As Any) As Long
Declare Function viIn8 Lib "VISA32.DLL" Alias "#273" (ByVal vi As Long, ByVal accSpace As Integer, ByVal offset As Long, val8 As Byte) As Long
Declare Function viOut8 Lib "VISA32.DLL" Alias "#274" (ByVal vi As Long, ByVal accSpace As Integer, ByVal offset As Long, ByVal val8 As Byte) As Long
Declare Function viIn16 Lib "VISA32.DLL" Alias "#261" (ByVal vi As Long, ByVal accSpace As Integer, ByVal offset As Long, val16 As Integer) As Long
Declare Function viOut16 Lib "VISA32.DLL" Alias "#262" (ByVal vi As Long, ByVal accSpace As Integer, ByVal offset As Long, ByVal val16 As Integer) As Long

(continues)

Table 3.2.2. Visual Basic Bindings for VISA Operations for the WINNT Framework (Continued)

```

Declare Function viIn32 Lib "VISA32.DLL" Alias "#281" (ByVal vi As Long, ByVal
    accSpace As Integer, ByVal offset As Long, val32 As Long)
    As Long
Declare Function viOut32 Lib "VISA32.DLL" Alias "#282" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    val32 As Long) As Long
Declare Function viMoveIn8 Lib "VISA32.DLL" Alias "#283" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    length As Long, buf8 As Byte) As Long
Declare Function viMoveOut8 Lib "VISA32.DLL" Alias "#284" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    length As Long, buf8 As Byte) As Long
Declare Function viMoveIn16 Lib "VISA32.DLL" Alias "#285" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    length As Long, buf16 As Integer) As Long
Declare Function viMoveOut16 Lib "VISA32.DLL" Alias "#286" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    length As Long, buf16 As Integer) As Long
Declare Function viMoveIn32 Lib "VISA32.DLL" Alias "#287" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    length As Long, buf32 As Long) As Long
Declare Function viMoveOut32 Lib "VISA32.DLL" Alias "#288" (ByVal vi As Long,
    ByVal accSpace As Integer, ByVal offset As Long, ByVal
    length As Long, buf32 As Long) As Long
Declare Function viMove Lib "VISA32.DLL" Alias "#200" (ByVal vi As Long, ByVal
    srcSpace As Integer, ByVal srcOffset As Long, ByVal
    srcWidth As Integer, ByVal destSpace As Integer, ByVal
    destOffset As Long, ByVal destWidth As Integer, ByVal
    srcLength As Long) As Long
Declare Function viMapAddress Lib "VISA32.DLL" Alias "#263" (ByVal vi As Long,
    ByVal mapSpace As Integer, ByVal mapOffset As Long, ByVal
    mapSize As Long, ByVal access As Integer, ByVal suggested
    As Long, address As Long) As Long
Declare Function viUnmapAddress Lib "VISA32.DLL" Alias "#264" (ByVal vi As
    Long) As Long
Declare Sub viPeek8 Lib "VISA32.DLL" Alias "#275" (ByVal vi As Long, ByVal
    address As Long, val8 As Byte)
Declare Sub viPoke8 Lib "VISA32.DLL" Alias "#276" (ByVal vi As Long, ByVal
    address As Long, ByVal val8 As Byte)
Declare Sub viPeek16 Lib "VISA32.DLL" Alias "#265" (ByVal vi As Long, ByVal
    address As Long, value16 As Integer)
Declare Sub viPoke16 Lib "VISA32.DLL" Alias "#266" (ByVal vi As Long, ByVal
    address As Long, ByVal value16 As Integer)
Declare Sub viPeek32 Lib "VISA32.DLL" Alias "#289" (ByVal vi As Long, ByVal
    address As Long, val32 As Long)
Declare Sub viPoke32 Lib "VISA32.DLL" Alias "#290" (ByVal vi As Long, ByVal
    address As Long, ByVal val32 As Long)
Declare Function viMemAlloc Lib "VISA32.DLL" Alias "#291" (ByVal vi As Long,
    ByVal memSize As Long, offset As Long) As Long

```

(continues)

Table 3.2.2. Visual Basic Bindings for VISA Operations for the WINNT Framework (Continued)

```

Declare Function viMemFree Lib "VISA32.DLL" Alias "#292" (ByVal vi As Long,
    ByVal offset As Long) As Long
Declare Function viGpibControlREN Lib "VISA32.DLL" Alias "#208" (ByVal vi As
    Long, ByVal mode As Integer) As Long
Declare Function viGpibControlATN Lib "VISA32.DLL" Alias "#210" (ByVal vi As
    Long, ByVal mode As Integer) As Long
Declare Function viGpibSendIFC Lib "VISA32.DLL" Alias "#211" (ByVal vi As
    Long) As Long
Declare Function viGpibCommand Lib "VISA32.DLL" Alias "#212" (ByVal vi As
    Long, ByVal Buffer As String, ByVal count As Long,
    retCount As Long) As Long
Declare Function viGpibPassControl Lib "VISA32.DLL" Alias "#213" (ByVal vi As
    Long, ByVal primAddr As Integer, ByVal secAddr As
    Integer) As Long
Declare Function viVxiCommandQuery Lib "VISA32.DLL" Alias "#209" (ByVal vi As
    Long, ByVal mode As Integer, ByVal devCmd As Long,
    devResponse As Long) As Long
Declare Function viAssertUtilSignal Lib "VISA32.DLL" Alias "#214" (ByVal vi As
    Long, ByVal line As Integer) As Long
Declare Function viAssertIntrSignal Lib "VISA32.DLL" Alias "#215" (ByVal vi As
    Long, ByVal mode As Integer, ByVal statusID As Long) As
    Long
Declare Function viMapTrigger Lib "VISA32.DLL" Alias "#216" (ByVal vi As Long,
    ByVal trigSrc As Integer, ByVal trigDest As Integer,
    ByVal mode As Integer) As Long
Declare Function viUnmapTrigger Lib "VISA32.DLL" Alias "#217" (ByVal vi As
    Long, ByVal trigSrc As Integer, ByVal trigDest As
    Integer) As Long
Declare Function viUsbControlOut Lib "VISA32.DLL" Alias "#293" (ByVal vi As
    Long, ByVal bmRequestType As Integer, ByVal bRequest As
    Integer, ByVal wValue As Integer, ByVal wIndex As
    Integer, ByVal wLength As Integer, buf As Byte) As Long
Declare Function viUsbControlIn Lib "VISA32.DLL" Alias "#294" (ByVal vi As
    Long, ByVal bmRequestType As Integer, ByVal bRequest As
    Integer, ByVal wValue As Integer, ByVal wIndex As
    Integer, ByVal wLength As Integer, buf As Byte, retCnt As
    Integer) As Long

```

RULE 3.2.6

All definitions specified in Table 3.2.2 for the WINNT framework **SHALL** be explicit within the `visa32.bas` file.

3.2.2 Operation Prototypes for WIN64 Framework**RULE 3.2.7**

Unless otherwise stated, all functions and operations specified in Table 3.2.1 **SHALL** be treated as fastcall when interfacing to the WIN64 Framework DLL.

RULE 3.2.8

The operations `viPrintf()`, `viScanf()`, and `viQueryf()` **SHALL** be treated as fastcall when interfacing to the WIN64 Framework DLL.

RULE 3.2.9

All pointers in Table 3.2.1 **SHALL** be treated as flat 64-bit pointers when interfacing to the WIN64 Framework DLL.

3.3 Completion and Error Codes

Table 3.3.1 lists the Completion and Error codes defined for all framework bindings.

Table 3.3.1. Completion and Error Codes

Completion and Error Codes	Values
VI_SUCCESS	0
VI_SUCCESS_EVENT_EN	3FFF0002h
VI_SUCCESS_EVENT_DIS	3FFF0003h
VI_SUCCESS_QUEUE_EMPTY	3FFF0004h
VI_SUCCESS_TERM_CHAR	3FFF0005h
VI_SUCCESS_MAX_CNT	3FFF0006h
VI_SUCCESS_DEV_NPRESENT	3FFF007Dh
VI_SUCCESS_TRIG_MAPPED	3FFF007Eh
VI_SUCCESS_QUEUE_NEMPTY	3FFF0080h
VI_SUCCESS_NCHAIN	3FFF0098h
VI_SUCCESS_NESTED_SHARED	3FFF0099h
VI_SUCCESS_NESTED_EXCLUSIVE	3FFF009Ah
VI_SUCCESS_SYNC	3FFF009Bh
VI_WARN_QUEUE_OVERFLOW	3FFF000Ch
VI_WARN_CONFIG_NLOADED	3FFF0077h
VI_WARN_NULL_OBJECT	3FFF0082h
VI_WARN_NSUP_ATTR_STATE	3FFF0084h
VI_WARN_UNKNOWN_STATUS	3FFF0085h
VI_WARN_NSUP_BUF	3FFF0088h
VI_WARN_EXT_FUNC_NIMPL	3FFF00A9h
VI_ERROR_SYSTEM_ERROR	BFFF0000h
VI_ERROR_INV_OBJECT	BFFF000Eh
VI_ERROR_INV_SESSION	BFFF000Eh
VI_ERROR_RSRC_LOCKED	BFFF000Fh
VI_ERROR_INV_EXPR	BFFF0010h
VI_ERROR_RSRC_NFOUND	BFFF0011h
VI_ERROR_INV_RSRC_NAME	BFFF0012h
VI_ERROR_INV_ACC_MODE	BFFF0013h
VI_ERROR_TMO	BFFF0015h
VI_ERROR_CLOSING_FAILED	BFFF0016h
VI_ERROR_INV_DEGREE	BFFF001Bh
VI_ERROR_INV_JOB_ID	BFFF001Ch

(continues)

Table 3.3.1. Completion and Error Codes (Continued)

Completion and Error Codes	Values
VI_ERROR_NSUP_ATTR	BFFF001Dh
VI_ERROR_NSUP_ATTR_STATE	BFFF001Eh
VI_ERROR_ATTR_READONLY	BFFF001Fh
VI_ERROR_INV_LOCK_TYPE	BFFF0020h
VI_ERROR_INV_ACCESS_KEY	BFFF0021h
VI_ERROR_INV_EVENT	BFFF0026h
VI_ERROR_INV_MECH	BFFF0027h
VI_ERROR_HNDLR_NINSTALLED	BFFF0028h
VI_ERROR_INV_HNDLR_REF	BFFF0029h
VI_ERROR_INV_CONTEXT	BFFF002Ah
VI_ERROR_NENABLED	BFFF002Fh
VI_ERROR_ABORT	BFFF0030h
VI_ERROR_RAW_WR_PROT_VIOL	BFFF0034h
VI_ERROR_RAW_RD_PROT_VIOL	BFFF0035h
VI_ERROR_OUTP_PROT_VIOL	BFFF0036h
VI_ERROR_INP_PROT_VIOL	BFFF0037h
VI_ERROR_BERR	BFFF0038h
VI_ERROR_IN_PROGRESS	BFFF0039h
VI_ERROR_INV_SETUP	BFFF003Ah
VI_ERROR_QUEUE_ERROR	BFFF003Bh
VI_ERROR_ALLOC	BFFF003Ch
VI_ERROR_INV_MASK	BFFF003Dh
VI_ERROR_IO	BFFF003Eh
VI_ERROR_INV_FMT	BFFF003Fh
VI_ERROR_NSUP_FMT	BFFF0041h
VI_ERROR_LINE_IN_USE	BFFF0042h
VI_ERROR_NSUP_MODE	BFFF0046h
VI_ERROR_SRQ_NOCCURRED	BFFF004Ah
VI_ERROR_INV_SPACE	BFFF004Eh
VI_ERROR_INV_OFFSET	BFFF0051h
VI_ERROR_INV_WIDTH	BFFF0052h
VI_ERROR_NSUP_OFFSET	BFFF0054h
VI_ERROR_NSUP_VAR_WIDTH	BFFF0055h
VI_ERROR_WINDOW_NMAPPED	BFFF0057h
VI_ERROR_RESP_PENDING	BFFF0059h

(continues)

Table 3.3.1. Completion and Error Codes (Continued)

Completion and Error Codes	Values
VI_ERROR_NLISTENERS	BFFF005Fh
VI_ERROR_NCIC	BFFF0060h
VI_ERROR_NSYS_CNTL	BFFF0061h
VI_ERROR_NSUP_OPER	BFFF0067h
VI_ERROR_INTR_PENDING	BFFF0068h
VI_ERROR_ASRL_PARITY	BFFF006Ah
VI_ERROR_ASRL_FRAMING	BFFF006Bh
VI_ERROR_ASRL_OVERRUN	BFFF006Ch
VI_ERROR_TRIG_NMAPPED	BFFF006Eh
VI_ERROR_NSUP_ALIGN_OFFSET	BFFF0070h
VI_ERROR_USER_BUF	BFFF0071h
VI_ERROR_RSRC_BUSY	BFFF0072h
VI_ERROR_NSUP_WIDTH	BFFF0076h
VI_ERROR_INV_PARAMETER	BFFF0078h
VI_ERROR_INV_PROT	BFFF0079h
VI_ERROR_INV_SIZE	BFFF007Bh
VI_ERROR_WINDOW_MAPPED	BFFF0080h
VI_ERROR_NIMPL_OPER	BFFF0081h
VI_ERROR_INV_LENGTH	BFFF0083h
VI_ERROR_INV_MODE	BFFF0091h
VI_ERROR_SESN_NLOCKED	BFFF009Ch
VI_ERROR_MEM_NSHARED	BFFF009Dh
VI_ERROR_LIBRARY_NFOUND	BFFF009Eh
VI_ERROR_NSUP_INTR	BFFF009Fh
VI_ERROR_INV_LINE	BFFF00A0h
VI_ERROR_FILE_ACCESS	BFFF00A1h
VI_ERROR_FILE_IO	BFFF00A2h
VI_ERROR_NSUP_LINE	BFFF00A3h
VI_ERROR_NSUP_MECH	BFFF00A4h
VI_ERROR_INTF_NUM_NCONFIG	BFFF00A5h
VI_ERROR_CONN_LOST	BFFF00A6h
VI_ERROR_NPERMISSION	BFFF00A8h

RULE 3.3.1

All Completion and Error codes specified in Table 3.3.1 **SHALL** be present in the `visa.h` and `visa32.bas` files.

RULE 3.3.2

The `visa.h` and `visa32.bas` files **SHALL** define all the Completion and Error codes to be the same bit pattern as those in Table 3.3.1.

OBSERVATION 3.3.1

Some ANSI C compilers may generate warnings when comparing signed and unsigned values. Since hexadecimal constants with the most significant bit set may be treated as unsigned values, comparing a variable of type `ViStatus` to any of the error codes could generate a warning. To avoid this situation, it is valid to represent the values in a different way. One example is to use their decimal equivalent (signed), which would normally not generate a warning.

OBSERVATION 3.3.2

Notice that all success and warning codes (Completion codes) have a value that is greater than or equal to 0, while all Error codes have a value that is less than 0. Therefore, an application determines whether an invocation of a given operation fails by checking to see whether the return value is *less than* 0 (as opposed to *not equal to* 0).

3.4 Attribute Values

Table 3.4.1 shows the attribute values used for all framework bindings.

Table 3.4.1. Attribute Values

Attribute Names	Values
VI_ATTR_RSRC_CLASS	BFFF0001h
VI_ATTR_RSRC_NAME	BFFF0002h
VI_ATTR_RSRC_IMPL_VERSION	3FFF0003h
VI_ATTR_RSRC_LOCK_STATE	3FFF0004h
VI_ATTR_MAX_QUEUE_LENGTH	3FFF0005h
VI_ATTR_USER_DATA	3FFF0007h
VI_ATTR_FDC_CHNL	3FFF000Dh
VI_ATTR_FDC_MODE	3FFF000Fh
VI_ATTR_FDC_GEN_SIGNAL_EN	3FFF0011h
VI_ATTR_FDC_USE_PAIR	3FFF0013h
VI_ATTR_SEND_END_EN	3FFF0016h
VI_ATTR_TERMCHAR	3FFF0018h
VI_ATTR_TMO_VALUE	3FFF001Ah
VI_ATTR_GPIB_READDR_EN	3FFF001Bh
VI_ATTR_IO_PROT	3FFF001Ch
VI_ATTR_DMA_ALLOW_EN	3FFF001Eh
VI_ATTR_ASRL_BAUD	3FFF0021h
VI_ATTR_ASRL_DATA_BITS	3FFF0022h
VI_ATTR_ASRL_PARITY	3FFF0023h
VI_ATTR_ASRL_STOP_BITS	3FFF0024h
VI_ATTR_ASRL_FLOW_CNTRL	3FFF0025h
VI_ATTR_RD_BUF_OPER_MODE	3FFF002Ah
VI_ATTR_RD_BUF_SIZE	3FFF002Bh
VI_ATTR_WR_BUF_OPER_MODE	3FFF002Dh
VI_ATTR_WR_BUF_SIZE	3FFF002Eh
VI_ATTR_SUPPRESS_END_EN	3FFF0036h
VI_ATTR_TERMCHAR_EN	3FFF0038h
VI_ATTR_DEST_ACCESS_PRIV	3FFF0039h
VI_ATTR_DEST_BYTE_ORDER	3FFF003Ah
VI_ATTR_SRC_ACCESS_PRIV	3FFF003Ch
VI_ATTR_SRC_BYTE_ORDER	3FFF003Dh
VI_ATTR_SRC_INCREMENT	3FFF0040h

(continues)

Table 3.4.1. Attribute Values (Continued)

Attribute Names	Values
VI_ATTR_DEST_INCREMENT	3FFF0041h
VI_ATTR_WIN_ACCESS_PRIV	3FFF0045h
VI_ATTR_WIN_BYTE_ORDER	3FFF0047h
VI_ATTR_GPIB_ATN_STATE	3FFF0057h
VI_ATTR_GPIB_ADDR_STATE	3FFF005Ch
VI_ATTR_GPIB_CIC_STATE	3FFF005Eh
VI_ATTR_GPIB_NDAC_STATE	3FFF0062h
VI_ATTR_GPIB_SRQ_STATE	3FFF0067h
VI_ATTR_GPIB_SYS_CNTRL_STATE	3FFF0068h
VI_ATTR_GPIB_HS488_CBL_LEN	3FFF0069h
VI_ATTR_CMDR_LA	3FFF006Bh
VI_ATTR_VXI_DEV_CLASS	3FFF006Ch
VI_ATTR_MAINFRAME_LA	3FFF0070h
VI_ATTR_MANF_NAME	BFFF0072h
VI_ATTR_MODEL_NAME	BFFF0077h
VI_ATTR_VXI_VME_INTR_STATUS	3FFF008Bh
VI_ATTR_VXI_TRIG_STATUS	3FFF008Dh
VI_ATTR_VXI_VME_SYSFAIL_STATE	3FFF0094h
VI_ATTR_WIN_BASE_ADDR	3FFF0098h
VI_ATTR_WIN_SIZE	3FFF009Ah
VI_ATTR_ASRL_AVAIL_NUM	3FFF00ACh
VI_ATTR_MEM_BASE	3FFF00ADh
VI_ATTR_ASRL_CTS_STATE	3FFF00AEh
VI_ATTR_ASRL_DCD_STATE	3FFF00AFh
VI_ATTR_ASRL_DSR_STATE	3FFF00B1h
VI_ATTR_ASRL_DTR_STATE	3FFF00B2h
VI_ATTR_ASRL_END_IN	3FFF00B3h
VI_ATTR_ASRL_END_OUT	3FFF00B4h
VI_ATTR_ASRL_REPLACE_CHAR	3FFF00BEh
VI_ATTR_ASRL_RI_STATE	3FFF00BFh
VI_ATTR_ASRL_RTS_STATE	3FFF00C0h
VI_ATTR_ASRL_XON_CHAR	3FFF00C1h
VI_ATTR_ASRL_XOFF_CHAR	3FFF00C2h
VI_ATTR_WIN_ACCESS	3FFF00C3h
VI_ATTR_RM_SESSION	3FFF00C4h

(continues)

Table 3.4.1. Attribute Values (Continued)

Attribute Names	Values
VI_ATTR_VXI_LA	3FFF00D5h
VI_ATTR_MANF_ID	3FFF00D9h
VI_ATTR_MEM_SIZE	3FFF00DDh
VI_ATTR_MEM_SPACE	3FFF00DEh
VI_ATTR_MODEL_CODE	3FFF00DFh
VI_ATTR_SLOT	3FFF00E8h
VI_ATTR_INTF_INST_NAME	BFFF00E9h
VI_ATTR_IMMEDIATE_SERV	3FFF0100h
VI_ATTR_INTF_PARENT_NUM	3FFF0101h
VI_ATTR_RSRC_SPEC_VERSION	3FFF0170h
VI_ATTR_INTF_TYPE	3FFF0171h
VI_ATTR_GPIB_PRIMARY_ADDR	3FFF0172h
VI_ATTR_GPIB_SECONDARY_ADDR	3FFF0173h
VI_ATTR_RSRC_MANF_NAME	BFFF0174h
VI_ATTR_RSRC_MANF_ID	3FFF0175h
VI_ATTR_INTF_NUM	3FFF0176h
VI_ATTR_TRIG_ID	3FFF0177h
VI_ATTR_GPIB_REN_STATE	3FFF0181h
VI_ATTR_GPIB_UNADDR_EN	3FFF0184h
VI_ATTR_DEV_STATUS_BYTE	3FFF0189h
VI_ATTR_FILE_APPEND_EN	3FFF0192h
VI_ATTR_VXI_TRIG_SUPPORT	3FFF0194h
VI_ATTR_TCPIP_ADDR	BFFF0195h
VI_ATTR_TCPIP_HOSTNAME	BFFF0196h
VI_ATTR_TCPIP_PORT	3FFF0197h
VI_ATTR_TCPIP_DEVICE_NAME	BFFF0199h
VI_ATTR_TCPIP_NODELAY	3FFF019Ah
VI_ATTR_TCPIP_KEEPALIVE	3FFF019Bh
VI_ATTR_4882_COMPLIANT	3FFF019Fh
VI_ATTR_USB_SERIAL_NUM	BFFF01A0h
VI_ATTR_USB_INTFC_NUM	3FFF01A1h
VI_ATTR_USB_PROTOCOL	3FFF01A7h
VI_ATTR_USB_MAX_INTR_SIZE	3FFF01AFh
VI_ATTR_JOB_ID	3FFF4006h
VI_ATTR_EVENT_TYPE	3FFF4010h

(continues)

Table 3.4.1. Attribute Values (Continued)

Attribute Names	Values
VI_ATTR_SIGP_STATUS_ID	3FFF4011h
VI_ATTR_RECV_TRIG_ID	3FFF4012h
VI_ATTR_INTR_STATUS_ID	3FFF4023h
VI_ATTR_STATUS	3FFF4025h
VI_ATTR_RET_COUNT	3FFF4026h
VI_ATTR_BUFFER	3FFF4027h
VI_ATTR_RECV_INTR_LEVEL	3FFF4041h
VI_ATTR_OPER_NAME	BFFF4042h
VI_ATTR_GPIB_RECV_CIC_STATE	3FFF4193h
VI_ATTR_RECV_TCPIP_ADDR	BFFF4198h
VI_ATTR_USB_RECV_INTR_SIZE	3FFF41B0h
VI_ATTR_USB_RECV_INTR_DATA	BFFF41B1h
VI_ATTR_PXI_DEV_NUM	3FFF0201h
VI_ATTR_PXI_FUNC_NUM	3FFF0202h
VI_ATTR_PXI_BUS_NUM	3FFF0205h
VI_ATTR_PXI_CHASSIS	3FFF0206h
VI_ATTR_PXI_SLOTPATH	BFFF0207h
VI_ATTR_PXI_SLOT_LBUS_LEFT	3FFF0208h
VI_ATTR_PXI_SLOT_LBUS_RIGHT	3FFF0209h
VI_ATTR_PXI_TRIG_BUS	3FFF020Ah
VI_ATTR_PXI_STAR_TRIG_BUS	3FFF020Bh
VI_ATTR_PXI_STAR_TRIG_LINE	3FFF020Ch
VI_ATTR_PXI_MEM_TYPE_BAR0	3FFF0211h
VI_ATTR_PXI_MEM_TYPE_BAR1	3FFF0212h
VI_ATTR_PXI_MEM_TYPE_BAR2	3FFF0213h
VI_ATTR_PXI_MEM_TYPE_BAR3	3FFF0214h
VI_ATTR_PXI_MEM_TYPE_BAR4	3FFF0215h
VI_ATTR_PXI_MEM_TYPE_BAR5	3FFF0216h
VI_ATTR_PXI_MEM_BASE_BAR0	3FFF0221h
VI_ATTR_PXI_MEM_BASE_BAR1	3FFF0222h
VI_ATTR_PXI_MEM_BASE_BAR2	3FFF0223h
VI_ATTR_PXI_MEM_BASE_BAR3	3FFF0224h
VI_ATTR_PXI_MEM_BASE_BAR4	3FFF0225h
VI_ATTR_PXI_MEM_BASE_BAR5	3FFF0226h
VI_ATTR_PXI_MEM_SIZE_BAR0	3FFF0231h

(continues)

Table 3.4.1. Attribute Values (Continued)

Attribute Names	Values
VI_ATTR_PXI_MEM_SIZE_BAR1	3FFF0232h
VI_ATTR_PXI_MEM_SIZE_BAR2	3FFF0233h
VI_ATTR_PXI_MEM_SIZE_BAR3	3FFF0234h
VI_ATTR_PXI_MEM_SIZE_BAR4	3FFF0235h
VI_ATTR_PXI_MEM_SIZE_BAR5	3FFF0236h
VI_ATTR_PXI_IS_EXPRESS	3FFF0240h
VI_ATTR_PXI_SLOT_LWIDTH	3FFF0241h
VI_ATTR_PXI_MAX_LWIDTH	3FFF0242h
VI_ATTR_PXI_ACTUAL_LWIDTH	3FFF0243h
VI_ATTR_PXI_DSTAR_BUS	3FFF0244h
VI_ATTR_PXI_DSTAR_SET	3FFF0245h
VI_ATTR_TCPIP_HISLIP_OVERLAP_EN	3FFF0300h
VI_ATTR_TCPIP_HISLIP_VERSION	3FFF0301h
VI_ATTR_TCPIP_HISLIP_MAX_MESSAGES_KB	3FFF0302h

RULE 3.4.1

All attribute codes specified in Table 3.4.1 **SHALL** appear in the `visa.h` and `visa32.bas` files.

RULE 3.4.2

The `visa.h` and `visa32.bas` files **SHALL** define all the attribute codes to be the same bit pattern as those in Table 3.4.1.

3.5 Event Type Values

Table 3.5.1 shows the event type values used for all framework bindings.

Table 3.5.1. Event Type Values

Attribute Names	Values
VI_EVENT_IO_COMPLETION	3FFF2009h
VI_EVENT_TRIG	BFFF200Ah
VI_EVENT_SERVICE_REQ	3FFF200Bh
VI_EVENT_CLEAR	3FFF200Dh
VI_EVENT_EXCEPTION	BFFF200Eh
VI_EVENT_GPIB_CIC	3FFF2012h
VI_EVENT_GPIB_TALK	3FFF2013h
VI_EVENT_GPIB_LISTEN	3FFF2014h
VI_EVENT_VXI_VME_SYSFAIL	3FFF201Dh
VI_EVENT_VXI_VME_SYSRESET	3FFF201Eh
VI_EVENT_VXI_SIGP	3FFF2020h
VI_EVENT_VXI_VME_INTR	BFFF2021h
VI_EVENT_TCPIP_CONNECT	3FFF2036h
VI_EVENT_USB_INTR	3FFF2037h
VI_EVENT_PXI_INTR	3FFF2022h
VI_ALL_ENABLED_EVENTS	3FFF7FFFh

RULE 3.5.1

All event types specified in Table 3.5.1 **SHALL** appear in the `visa.h` and `visa32.bas` files.

RULE 3.5.2

The `visa.h` and `visa32.bas` files **SHALL** define all the event types to be the same bit pattern as those in Table 3.5.1.

3.6 Values and Ranges

Table 3.6.1 shows the values used in all framework bindings.

Table 3.6.1. Values and Ranges

Name	Value
VI_FIND_BUFLEN	256
VI_NULL	0
VI_TRUE	1
VI_FALSE	0
VI_INTF_GPIB	1
VI_INTF_VXI	2
VI_INTF_GPIB_VXI	3
VI_INTF_ASRL	4
VI_INTF_TCPIP	6
VI_NORMAL	1
VI_FDC	2
VI_HS488	3
VI_ASRL488	4
VI_FDC_NORMAL	1
VI_FDC_STREAM	2
VI_A16_SPACE	1
VI_A24_SPACE	2
VI_A32_SPACE	3
VI_UNKNOWN_SLOT	-1
VI_UNKNOWN_LA	-1
VI_UNKNOWN_LEVEL	-1
VI_QUEUE	1
VI_HNDLR	2
VI_SUSPEND_HNDLR	4
VI_ALL_MECH	FFFFh
VI_ANY_HNDLR	0
VI_TRIG_SW	-1
VI_TRIG_TTL0	0
VI_TRIG_TTL1	1
VI_TRIG_TTL2	2
VI_TRIG_TTL3	3
VI_TRIG_TTL4	4

Name	Value
VI_TRIG_TTL5	5
VI_TRIG_TTL6	6
VI_TRIG_TTL7	7
VI_TRIG_ECL0	8
VI_TRIG_ECL1	9
VI_TRIG_PANEL_IN	27
VI_TRIG_PANEL_OUT	28
VI_TRIG_PROT_DEFAULT	0
VI_TRIG_PROT_ON	1
VI_TRIG_PROT_OFF	2
VI_TRIG_PROT_SYNC	5
VI_READ_BUF	1
VI_WRITE_BUF	2
VI_READ_BUF_DISCARD	4
VI_WRITE_BUF_DISCARD	8
VI_ASRL_IN_BUF	16
VI_ASRL_OUT_BUF	32
VI_ASRL_IN_BUF_DISCARD	64
VI_ASRL_OUT_BUF_DISCARD	128
VI_FLUSH_ON_ACCESS	1
VI_FLUSH_WHEN_FULL	2
VI_FLUSH_DISABLE	3
VI_NMAPPED	1
VI_USE_OPERS	2
VI_DEREF_ADDR	3
VI_TMO_IMMEDIATE	0
VI_TMO_INFINITE	FFFFFFFFh
VI_NO_LOCK	0
VI_EXCLUSIVE_LOCK	1
VI_SHARED_LOCK	2
VI_LOAD_CONFIG	4
VI_NO_SEC_ADDR	FFFFh

(continues)

Table 3.6.1. Values and Ranges (Continued)

Name	Value
VI_ASRL_PAR_NONE	0
VI_ASRL_PAR_ODD	1
VI_ASRL_PAR_EVEN	2
VI_ASRL_PAR_MARK	3
VI_ASRL_PAR_SPACE	4
VI_ASRL_STOP_ONE	10
VI_ASRL_STOP_ONE5	15
VI_ASRL_STOP_TWO	20
VI_ASRL_FLOW_NONE	0
VI_ASRL_FLOW_XON_XOFF	1
VI_ASRL_FLOW_RTS_CTS	2
VI_ASRL_FLOW_DTR_DSR	4
VI_ASRL_END_NONE	0
VI_ASRL_END_LAST_BIT	1
VI_ASRL_END_TERMCHAR	2
VI_ASRL_END_BREAK	3
VI_BIG_ENDIAN	0
VI_LITTLE_ENDIAN	1
VI_WIDTH_8	1
VI_WIDTH_16	2
VI_WIDTH_32	4
VI_STATE_ASSERTED	1
VI_STATE_UNASSERTED	0
VI_STATE_UNKNOWN	-1
VI_GPIB_HS488_DISABLED	0
VI_GPIB_HS488_NIMPL	-1
VI_VXI_CLASS_MEMORY	0
VI_VXI_CLASS_EXTENDED	1
VI_VXI_CLASS_MESSAGE	2
VI_VXI_CLASS_REGISTER	3
VI_VXI_CLASS_OTHER	4
VI_UTIL_ASSERT_SYSRESET	1
VI_UTIL_ASSERT_SYSFAIL	2
VI_UTIL_DEASSERT_SYSFAIL	3
VI_TRIG_ALL	-2

Name	Value
VI_DATA_PRIV	0
VI_DATA_NPRIV	1
VI_PROG_PRIV	2
VI_PROG_NPRIV	3
VI_BLKCK_PRIV	4
VI_BLKCK_NPRIV	5
VI_D64_PRIV	6
VI_D64_NPRIV	7
VI_LOCAL_SPACE	0
VI_GPIB_REN_DEASSERT	0
VI_GPIB_REN_ASSERT	1
VI_GPIB_REN_DEASSERT_GTL	2
VI_GPIB_REN_ASSERT_ADDRESS	3
VI_GPIB_REN_ASSERT_LLO	4
VI_GPIB_REN_ASSERT_ADDRESS_LLO	5
VI_GPIB_REN_ADDRESS_GTL	6
VI_VXI_CMD16	0200h
VI_VXI_CMD16_RESP16	0202h
VI_VXI_RESP16	0002h
VI_VXI_CMD32	0400h
VI_VXI_CMD32_RESP16	0402h
VI_VXI_CMD32_RESP32	0404h
VI_VXI_RESP32	0004h
VI_GPIB_ATN_DEASSERT	0
VI_GPIB_ATN_ASSERT	1
VI_GPIB_ATN_DEASSERT_HANDSHAKE	2
VI_GPIB_ATN_ASSERT_IMMEDIATE	3
VI_ASSERT_SIGNAL	-1
VI_ASSERT_USE_ASSIGNED	0
VI_ASSERT_IRQ1	1
VI_ASSERT_IRQ2	2
VI_ASSERT_IRQ3	3
VI_ASSERT_IRQ4	4
VI_ASSERT_IRQ5	5
VI_ASSERT_IRQ6	6

(continues)

Table 3.6.1. Values and Ranges (Continued)

Name	Value
VI_ASSERT_IRQ7	7
VI_GPIB_UNADDRESSED	0
VI_GPIB_TALKER	1
VI_GPIB_LISTENER	2
VI_INTF_USB	7
VI_PROT_FDC	2
VI_PROT_4882_STRS	4
VI_OPAQUE_SPACE	FFFFh
VI_INTF_PXI	5
VI_PXI_ALLOC_SPACE	9
VI_PXI_CFG_SPACE	10
VI_PXI_BAR0_SPACE	11
VI_PXI_BAR1_SPACE	12
VI_PXI_BAR2_SPACE	13
VI_PXI_BAR3_SPACE	14
VI_PXI_BAR4_SPACE	15
VI_PXI_BAR5_SPACE	16
VI_PXI_ADDR_NONE	0
VI_PXI_ADDR_MEM	1
VI_PXI_ADDR_IO	2
VI_PXI_ADDR_CFG	3
VI_A64_SPACE	4
VI_WIDTH_64	8

Name	Value
VI_IO_IN_BUF	16
VI_IO_OUT_BUF	32
VI_IO_IN_BUF_DISCARD	64
VI_IO_OUT_BUF_DISCARD	128
VI_PROT_NORMAL	1
VI_PROT_HS488	3
VI_PROT_USBTMC_VENDOR	5
VI_UNKNOWN_CHASSIS	-1
VI_UNKNOWN_TRIG	-1
VI_PXI_LBUS_STAR_TRIG_BUS_0	1000
VI_PXI_LBUS_STAR_TRIG_BUS_1	1001
VI_PXI_LBUS_STAR_TRIG_BUS_2	1002
VI_PXI_LBUS_STAR_TRIG_BUS_3	1003
VI_PXI_LBUS_STAR_TRIG_BUS_4	1004
VI_PXI_LBUS_STAR_TRIG_BUS_5	1005
VI_PXI_LBUS_STAR_TRIG_BUS_6	1006
VI_PXI_LBUS_STAR_TRIG_BUS_7	1007
VI_PXI_LBUS_STAR_TRIG_BUS_8	1008
VI_PXI_LBUS_STAR_TRIG_BUS_9	1009
VI_PXI_STAR_TRIG_CONTROLLER	1413
VI_TRIG_PROT_RESERVE	6
VI_TRIG_PROT_UNRESERVE	7

RULE 3.6.1

All values and ranges specified in Table 3.6.1 **SHALL** appear in the `visa.h` and `visa32.bas` files.

RULE 3.6.2

The `visa.h` and `visa32.bas` files **SHALL** define all the values and ranges to be the same bit pattern as those in Table 3.6.1.

RULE 3.6.3

The range of the attribute `VI_ATTR_USER_DATA` **SHALL** be 0 to FFFFFFFFh.

3.7 Library Requirements

These sections discuss issues with the framework libraries and show the procedure definition exports for all framework bindings.

RULE 3.7.1

The library containing VISA **SHALL** be dynamically loadable.

3.7.1 Library Requirements for WINNT and WIN64 Frameworks

Table 3.7.1 shows the procedure definition exports for the WINNT and WIN64 Frameworks.

Table 3.7.1. Procedure Definition Exports for the WINNT and WIN64 Frameworks

Entry Point	Ordinal Number
viGetDefaultRM	128
viOpenDefaultRM	141
viFindRsrc	129
viFindNext	130
viOpen	131
viLock	144
viUnlock	145
viEnableEvent	135
viDisableEvent	136
viDiscardEvents	137
viWaitOnEvent	138
viInstallHandler	139
viUninstallHandler	140
viMove	200
viMoveAsync	201
viBufWrite	202
viBufRead	203
viSprintf	204
viVSprintf	205
viSScanf	206
viVSScanf	207
viGpibControlREN	208
viVxiCommandQuery	209

Entry Point	Ordinal Number
viClose	132
viGetAttribute	133
viSetAttribute	134
viStatusDesc	142
viTerminate	143
viReadSTB	259
viClear	260
viSetBuf	267
viFlush	268
viPrintf	269
viVPrintf	270
viScanf	271
viVScanf	272
viQueryf	279
viVQueryf	280
viIn8	273
viOut8	274
viIn16	261
viOut16	262
viIn32	281
viOut32	282
viMoveIn8	283
viMoveOut8	284

(continues)

Table 3.7.1. Procedure Definition Exports for the WINNT and WIN64 Frameworks (Continued)

Entry Point	Ordinal Number
viRead	256
viReadAsync	277
viWrite	257
viWriteAsync	278
viAssertTrigger	258
viMapAddress	263
viUnmapAddress	264
viMemAlloc	291
viMemFree	292
viGpibControlATN	210
viGpibSendIFC	211
viGpibCommand	212
viGpibPassControl	213
viAssertUtilSignal	214
viAssertIntrSignal	215
viParseRsrcEx	147
viUsbControlIn	294
viOut64	221
viOut8Ex	223
viOut16Ex	225
viOut32Ex	227
viOut64Ex	229
viMoveOut64	231
viMoveOut8Ex	233
viMoveOut16Ex	235
viMoveOut32Ex	237
viMoveOut64Ex	239
viMoveAsyncEx	241
viMemAllocEx	243
viPeek64	245

Entry Point	Ordinal Number
viMoveIn16	285
viMoveOut16	286
viMoveIn32	287
viMoveOut32	288
viPeek8	275
viPoke8	276
viPeek16	265
viPoke16	266
viPeek32	289
viPoke32	290
viParseRsrc	146
viMapTrigger	216
viUnmapTrigger	217
viWriteFromFile	218
viReadToFile	219
viUsbControlOut	293
viIn64	220
viIn8Ex	222
viIn16Ex	224
viIn32Ex	226
viIn64Ex	228
viMoveIn64	230
viMoveIn8Ex	232
viMoveIn16Ex	234
viMoveIn32Ex	236
viMoveIn64Ex	238
viMoveEx	240
viMapAddressEx	242
viMemFreeEx	244
viPoke64	246

RULE 3.7.2

The WINNT Framework DLL **SHALL** be named `visa32.dll`.

RULE 3.7.3

The WINNT Framework DLL **SHALL** be a 32-bit DLL.

RULE 3.7.4

The WINNT Framework DLL **SHALL** use the exports in the procedure definition file (`visa32.def` file) specified in Table 3.7.1.

RULE 3.7.5

The WIN64 Framework DLL **SHALL** be named `visa64.dll`.

RULE 3.7.6

The WIN64 Framework DLL **SHALL** be a 64-bit DLL.

RULE 3.7.7

The WIN64 Framework DLL **SHALL** use the exports in the procedure definition file (`visa64.def` file) specified in Table 3.7.1.

OBSERVATION 3.7.1

The location where the VISA library is installed is specified in VPP-6.

3.8 Miscellaneous

RULE 3.8.1

Every VISA 4.0 implementation **SHALL** provide the following `#define` in the `visa.h` file:

```
#define VI_SPEC_VERSION (0x00400000UL)
```

RULE 3.8.2

Every VISA 4.0 implementation **SHALL** provide the following constant in the `visa32.bas` file:

```
Global Const VI_SPEC_VERSION = &H00400000&
```

RULE 3.8.3

The default contents (with no user-defined macros enabled) of the compiled or interpreted versions of the `visatype.h`, `visa.h`, and `visa32.bas` files **SHALL** be exactly the same as the compiled or interpreted versions of the corresponding files listed in Appendix A, *Implementation Files*, of this document.

PERMISSION 3.8.1

A vendor **MAY** provide conditionally compiled or interpreted extensions to the `visatype.h`, `visa.h`, and `visa32.bas` files listed in Appendix A, *Implementation Files*, of this document.

PERMISSION 3.8.2

Any vendor-specific extension to the `visatype.h`, `visa.h`, and `visa32.bas` files **MAY** be either binary compatible or non-interoperable.

RULE 3.8.4

Binary-compatible vendor-specific extensions **SHALL** be enabled via a user-defined macro of the form `PREFIX_<extension>`.

RULE 3.8.5

Non-interoperable vendor-specific extensions **SHALL** be enabled via a user-defined macro of the form `PREFIX_NONINTEROP_<extension>`.

RULE 3.8.6

The `PREFIX` used in Rules 3.8.4 and 3.8.5 **SHALL** begin with two characters based on the instrument vendor as defined in VPP-9, *Instrument Vendor Abbreviations*, followed by the characters `VISA`.

OBSERVATION 3.8.1

Rule 3.8.3 through 3.8.6 and Permissions 3.8.1 and 3.8.2 allow for vendor-specific extensions, provided that the default version (with no user-defined macros enabled) compiles to the same output as the files provided in this specification. Rule 3.8.3 provides for multi-vendor interoperability for *VXIplug&play* applications and instrument drivers compiled without user-defined macros.

OBSERVATION 3.8.2

Two examples of a valid PREFIX, as specified in Rule 3.8.6, are NIVISA and HPVISA.

PERMISSION 3.8.3

A vendor **MAY** use either one <extension> user-defined macro to cover all extensions, or a unique <extension> macro for each extension.

OBSERVATION 3.8.3

An example of a non-interoperable extension is the addition of an operation not published in this specification. An application using that operation may behave incorrectly or even fail to run if used with a different vendor's VISA implementation that does not have that operation.

RECOMMENDATION 3.8.1

Non-compatible extensions to the `visatype.h`, `visa.h`, and `visa32.bas` files should provide a warning when such a feature is enabled.

OBSERVATION 3.8.4

Notice that not all compilers or interpreters can produce warning messages.

RULE 3.8.7

IF a vendor-specific extension overrides any operation, attribute, or other feature provided by the VISA specification, **THEN** the vendor providing that extension **SHALL** ensure that the feature is binary compatible with other vendors' implementations of VISA.

RULE 3.8.8

Every VISA implementation **SHALL** follow the VISA installation requirements as listed in VPP-6, *Installation and Packaging Specification*.

Table 3.8.1. Bit Pattern for Attributes

		← 14								← 12			
31	30	29	...	16	15	14	13	12	11	...	0		

Bit 31: Pass by value or by reference

0 = by value

1 = by reference

Bit 30: Reserved (always 0)

Bits 29-16: Manufacturer ID

0-0FFF = VXI defined

1000-3FFF = VXi*plug&play* defined

- 3FFC = instrument drivers

- 3FFF = VISA

Bit 15: Published or internal attribute

0 = published

1 = internal / undocumented

Bit 14: Attribute class association

0 = defined by the VISA template or an individual resource

1 = defined by an event

Bits 13-12: Reserved (always 0)

Bits 11-0: Unique sequence value

Table 3.8.2. Bit Pattern for Status Codes

		← 14								← 12			
31	30	29	...	16	15	14	13	12	11	...	0		

Bit 31: Success or failure

0 = success or warning

1 = error

Bit 30: Reserved (always 0)

Bits 29-16: Manufacturer ID

0-0FFF = VXI defined

1000-3FFF = VXi*plug&play* defined

- 3FFC = instrument drivers

- 3FFF = VISA

Bit 15: Published or internal status code

0 = published

1 = internal / undocumented

Bits 14-12: Reserved (always 0)

Bits 11-0: Unique sequence value

RULE 3.8.9

IF a vendor-specific extension includes attributes or status codes, **THEN** the numbers for those attributes and status codes **SHALL** be consistent with the coding scheme presented in Tables 3.8.1 and 3.8.2.

OBSERVATION 3.8.5

All VISA-defined attributes and status codes listed in Tables 3.3.1 and 3.4.1 are consistent with the coding scheme presented in Tables 3.8.1 and 3.8.2.

Appendix A Implementation Files

A.1 Contents of visatype.h File

This file reflects the required implementation of the specifications given in this document.

```

/*-----*/
/* Distributed by VXIplug&play Systems Alliance */
/* */
/* Do not modify the contents of this file. */
/*-----*/
/* */
/* Title : VISATYPE.H */
/* Date : 04-14-2006 */
/* Purpose : Fundamental VISA data types and macro definitions */
/* */
/*-----*/

#ifndef __VISATYPE_HEADER__
#define __VISATYPE_HEADER__

#if defined(_WIN64)
#define _VI_FAR
#define _VI_FUNC __fastcall
#define _VI_FUNC__ __fastcall
#define _VI_FUNC__H __fastcall
#define _VI_SIGNED signed
#elif defined(WIN32) || defined(_WIN32) || defined(__NT__) && !defined(_NI_mswin16_)
#define _VI_FAR
#define _VI_FUNC __stdcall
#define _VI_FUNC__ _cdecl
#define _VI_FUNC__H __stdcall
#define _VI_SIGNED signed
#elif defined(_CVI_) && defined(_NI_i386_)
#define _VI_FAR
#define _VI_FUNC _pascal
#define _VI_FUNC__ _pascal
#define _VI_FUNC__H _pascal
#define _VI_SIGNED signed
#elif defined(_WINDOWS) || defined(_Windows) && !defined(_NI_mswin16_)
#define _VI_FAR _far
#define _VI_FUNC _far _pascal _export
#define _VI_FUNC__ _far _cdecl _export
#define _VI_FUNC__H _far _pascal
#define _VI_SIGNED signed
#elif defined(hpux) || defined(__hpux) && (defined(__cplusplus) || defined(__cplusplus__))
#define _VI_FAR
#define _VI_FUNC
#define _VI_FUNC__
#define _VI_FUNC__H
#define _VI_SIGNED signed
#else
#define _VI_FAR
#define _VI_FUNC
#define _VI_FUNC__
#define _VI_FUNC__H
#define _VI_SIGNED signed
#endif

#define _VI_ERROR (-2147483647L-1) /* 0x80000000 */
#define _VI_PTR _VI_FAR *

```

```

/*- VISA Types -----*/

#ifndef _VI_INT64_UINT64_DEFINED
#if defined(_WIN64) || ((defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
defined(__NT__)) && !defined(_NI_mswin16_))
#if (defined(_MSC_VER) && (_MSC_VER >= 1200)) || (defined(_CVI_) && (_CVI_ >= 700)) ||
(defined(__BORLANDC__) && (__BORLANDC__ >= 0x0520))
typedef unsigned __int64 ViUInt64;
typedef _VI_SIGNED __int64 ViInt64;
#define _VI_INT64_UINT64_DEFINED
#endif
#endif
#define _VISA_ENV_IS_64_BIT
#else
/* This is a 32-bit OS, not a 64-bit OS */
#endif
#endif
#elif defined(__GNUC__) && (__GNUC__ >= 3)
#include <limits.h>
#include <sys/types.h>
typedef u_int64_t ViUInt64;
typedef int64_t ViInt64;
#define _VI_INT64_UINT64_DEFINED
#if defined(LONG_MAX) && (LONG_MAX > 0x7FFFFFFFL)
#define _VISA_ENV_IS_64_BIT
#else
/* This is a 32-bit OS, not a 64-bit OS */
#endif
#else
/* This platform does not support 64-bit types */
#endif
#endif

#if defined(_VI_INT64_UINT64_DEFINED)
typedef ViUInt64 _VI_PTR ViPUInt64;
typedef ViUInt64 _VI_PTR ViAUInt64;
typedef ViInt64 _VI_PTR ViPInt64;
typedef ViInt64 _VI_PTR ViAInt64;
#endif

typedef unsigned long ViUInt32;
typedef ViUInt32 _VI_PTR ViPUInt32;
typedef ViUInt32 _VI_PTR ViAUInt32;

typedef _VI_SIGNED long ViInt32;
typedef ViInt32 _VI_PTR ViPInt32;
typedef ViInt32 _VI_PTR ViAInt32;

typedef unsigned short ViUInt16;
typedef ViUInt16 _VI_PTR ViPUInt16;
typedef ViUInt16 _VI_PTR ViAUInt16;

typedef _VI_SIGNED short ViInt16;
typedef ViInt16 _VI_PTR ViPInt16;
typedef ViInt16 _VI_PTR ViAInt16;

typedef unsigned char ViUInt8;
typedef ViUInt8 _VI_PTR ViPUInt8;
typedef ViUInt8 _VI_PTR ViAUInt8;

typedef _VI_SIGNED char ViInt8;
typedef ViInt8 _VI_PTR ViPInt8;
typedef ViInt8 _VI_PTR ViAInt8;

typedef char ViChar;
typedef ViChar _VI_PTR ViPChar;
typedef ViChar _VI_PTR ViAChar;

typedef unsigned char ViByte;
typedef ViByte _VI_PTR ViPByte;
typedef ViByte _VI_PTR ViAByte;

typedef void _VI_PTR ViAddr;
typedef ViAddr _VI_PTR ViPAddr;
typedef ViAddr _VI_PTR ViAAddr;

typedef float ViReal32;
typedef ViReal32 _VI_PTR ViPReal32;

```

```

typedef ViReal32    _VI_PTR ViAReal32;

typedef double      ViReal64;
typedef ViReal64   _VI_PTR ViPReal64;
typedef ViReal64   _VI_PTR ViAReal64;

typedef ViPByte     ViBuf;
typedef ViPByte     ViPBuf;
typedef ViPByte     _VI_PTR ViABuf;

typedef ViPChar     ViString;
typedef ViPChar     ViPString;
typedef ViPChar     _VI_PTR ViAString;

typedef ViString    ViRsrc;
typedef ViString    ViPRsrc;
typedef ViString    _VI_PTR ViARsrc;

typedef ViUInt16    ViBoolean;
typedef ViBoolean   _VI_PTR ViPBoolean;
typedef ViBoolean   _VI_PTR ViABoolean;

typedef ViInt32     ViStatus;
typedef ViStatus    _VI_PTR ViPStatus;
typedef ViStatus    _VI_PTR ViAStatus;

typedef ViUInt32    ViVersion;
typedef ViVersion   _VI_PTR ViPVersion;
typedef ViVersion   _VI_PTR ViAVersion;

typedef ViUInt32    ViObject;
typedef ViObject    _VI_PTR ViPObject;
typedef ViObject    _VI_PTR ViAObject;

typedef ViObject    ViSession;
typedef ViSession   _VI_PTR ViPSession;
typedef ViSession   _VI_PTR ViASession;

typedef ViUInt32    ViAttr;

#ifndef _VI_CONST_STRING_DEFINED
typedef const ViChar * ViConstString;
#define _VI_CONST_STRING_DEFINED
#endif

/*- Completion and Error Codes -----*/
#define VI_SUCCESS      (0L)

/*- Other VISA Definitions -----*/
#define VI_NULL         (0)
#define VI_TRUE         (1)
#define VI_FALSE        (0)

/*- Backward Compatibility Macros -----*/
#define VISAFN          _VI_FUNC
#define ViPtr           _VI_PTR

#endif

/*- The End -----*/

```

A.2 Contents of visa.h File

This file reflects the required implementation of the specifications given in this document.

```

/*-----*/
/* Distributed by VXIplug&play Systems Alliance */
/* */
/* Do not modify the contents of this file. */
/*-----*/
/* */
/* Title : VISA.H */
/* Date : 06-08-2010 */
/* Purpose : Include file for the VISA Library 5.0 specification */
/*-----*/

#ifndef __VISA_HEADER__
#define __VISA_HEADER__

#include <stdarg.h>
#if !defined(__VISATYPE_HEADER__)
#include "visatype.h"
#endif

#define VI_SPEC_VERSION (0x00500000UL)

#if defined(__cplusplus) || defined(__cplusplus__)
extern "C" {
#endif

#if defined(_CVI_)
#pragma EnableLibraryRuntimeChecking
#endif

/*- VISA Types -----*/
typedef ViObject ViEvent;
typedef ViEvent _VI_PTR ViPEvent;
typedef ViObject ViFindList;
typedef ViFindList _VI_PTR ViPFindList;

#if defined(_VI_INT64_UINT64_DEFINED) && defined(_VISA_ENV_IS_64_BIT)
typedef ViUInt64 ViBusAddress;
typedef ViUInt64 ViBusSize;
typedef ViUInt64 ViAttrState;
#else
typedef ViUInt32 ViBusAddress;
typedef ViUInt32 ViBusSize;
typedef ViUInt32 ViAttrState;
#endif

#if defined(_VI_INT64_UINT64_DEFINED)
typedef ViUInt64 ViBusAddress64;
typedef ViBusAddress64 _VI_PTR ViPBusAddress64;
#endif

typedef ViUInt32 ViEventType;
typedef ViEventType _VI_PTR ViPEventType;
typedef ViEventType _VI_PTR ViAEventType;
typedef void _VI_PTR ViPAttrState;
typedef ViAttr _VI_PTR ViPAttr;
typedef ViAttr _VI_PTR ViAAttr;

typedef ViString ViKeyId;
typedef ViPString ViPKeyId;
typedef ViUInt32 ViJobId;
typedef ViJobId _VI_PTR ViPJobId;
typedef ViUInt32 ViAccessMode;
typedef ViAccessMode _VI_PTR ViPAccessMode;
typedef ViBusAddress _VI_PTR ViPBusAddress;
typedef ViUInt32 ViEventFilter;

typedef va_list ViVAList;

typedef ViStatus (_VI_FUNC _VI_PTR ViHndlr)

```

```

    (ViSession vi, ViEventType eventType, ViEvent event, ViAddr userHandle);

/*- Resource Manager Functions and Operations -----*/
ViStatus _VI_FUNC viOpenDefaultRM (ViPSession vi);
ViStatus _VI_FUNC viFindRsrc      (ViSession sesn, ViString expr, ViPFindList vi,
ViStatus _VI_FUNC viFindNext      (ViFindList vi, ViChar _VI_FAR desc[]);
ViStatus _VI_FUNC viParseRsrc     (ViSession rmSesn, ViRsrc rsrcName,
ViStatus _VI_FUNC viParseRsrcEx   (ViSession rmSesn, ViRsrc rsrcName, ViPUInt16 intfType,
ViStatus _VI_FUNC viOpen          (ViSession sesn, ViRsrc name, ViAccessMode mode,
ViStatus _VI_FUNC viOpen          (ViSession sesn, ViRsrc name, ViAccessMode mode,
ViUInt32 timeout, ViPSession vi);

/*- Resource Template Operations -----*/
ViStatus _VI_FUNC viClose         (ViObject vi);
ViStatus _VI_FUNC viSetAttribute  (ViObject vi, ViAttr attrName, ViAttrState attrValue);
ViStatus _VI_FUNC viGetAttribute  (ViObject vi, ViAttr attrName, void _VI_PTR attrValue);
ViStatus _VI_FUNC viStatusDesc   (ViObject vi, ViStatus status, ViChar _VI_FAR desc[]);
ViStatus _VI_FUNC viTerminate    (ViObject vi, ViUInt16 degree, ViJobId jobId);

ViStatus _VI_FUNC viLock         (ViSession vi, ViAccessMode lockType, ViUInt32 timeout,
ViStatus _VI_FUNC viUnlock       (ViSession vi);
ViStatus _VI_FUNC viEnableEvent  (ViSession vi, ViEventType eventType, ViUInt16 mechanism,
ViStatus _VI_FUNC viDisableEvent (ViSession vi, ViEventType eventType, ViUInt16 mechanism);
ViStatus _VI_FUNC viDiscardEvents (ViSession vi, ViEventType eventType, ViUInt16 mechanism);
ViStatus _VI_FUNC viWaitOnEvent  (ViSession vi, ViEventType inEventType, ViUInt32 timeout,
ViStatus _VI_FUNC viInstallHandler (ViSession vi, ViEventType eventType, ViHndlr handler,
ViStatus _VI_FUNC viUninstallHandler (ViSession vi, ViEventType eventType, ViHndlr handler,
ViAddr userHandle);

/*- Basic I/O Operations -----*/
ViStatus _VI_FUNC viRead         (ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus _VI_FUNC viReadAsync   (ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPJobId jobId);
ViStatus _VI_FUNC viReadToFile  (ViSession vi, ViConstString filename, ViUInt32 cnt,
ViStatus _VI_FUNC viWrite       (ViSession vi, ViBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus _VI_FUNC viWriteAsync  (ViSession vi, ViBuf buf, ViUInt32 cnt, ViPJobId jobId);
ViStatus _VI_FUNC viWriteFromFile (ViSession vi, ViConstString filename, ViUInt32 cnt,
ViStatus _VI_FUNC viAssertTrigger (ViSession vi, ViUInt16 protocol);
ViStatus _VI_FUNC viReadSTB     (ViSession vi, ViPUInt16 status);
ViStatus _VI_FUNC viClear       (ViSession vi);

/*- Formatted and Buffered I/O Operations -----*/
ViStatus _VI_FUNC viSetBuf      (ViSession vi, ViUInt16 mask, ViUInt32 size);
ViStatus _VI_FUNC viFlush       (ViSession vi, ViUInt16 mask);

ViStatus _VI_FUNC viBufWrite    (ViSession vi, ViBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus _VI_FUNC viBufRead     (ViSession vi, ViPBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);

ViStatus _VI_FUNC viPrintf      (ViSession vi, ViString writeFmt, ...);
ViStatus _VI_FUNC viVPrintf     (ViSession vi, ViString writeFmt, ViVAlList params);
ViStatus _VI_FUNC viSPrintf     (ViSession vi, ViPBuf buf, ViString writeFmt, ...);
ViStatus _VI_FUNC viVSPrintf    (ViSession vi, ViPBuf buf, ViString writeFmt,
ViStatus _VI_FUNC viScanf       (ViSession vi, ViString readFmt, ...);
ViStatus _VI_FUNC viVScanf      (ViSession vi, ViString readFmt, ViVAlList params);
ViStatus _VI_FUNC viSScanf      (ViSession vi, ViBuf buf, ViString readFmt, ...);
ViStatus _VI_FUNC viVSScanf     (ViSession vi, ViBuf buf, ViString readFmt,
ViStatus _VI_FUNC viQueryf      (ViSession vi, ViString writeFmt, ViString readFmt, ...);
ViStatus _VI_FUNC viVQueryf     (ViSession vi, ViString writeFmt, ViString readFmt,

```

```

/*- Memory I/O Operations -----*/

ViStatus _VI_FUNC viIn8      (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViPUInt8 val8);
ViStatus _VI_FUNC viOut8     (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViUInt8 val8);
ViStatus _VI_FUNC viIn16     (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViPUInt16 val16);
ViStatus _VI_FUNC viOut16    (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViUInt16 val16);
ViStatus _VI_FUNC viIn32     (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViPUInt32 val32);
ViStatus _VI_FUNC viOut32    (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViUInt32 val32);

#if defined(_VI_INT64_UINT64_DEFINED)
ViStatus _VI_FUNC viIn64     (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViPUInt64 val64);
ViStatus _VI_FUNC viOut64    (ViSession vi, ViUInt16 space,
                             ViBusAddress offset, ViUInt64 val64);

ViStatus _VI_FUNC viIn8Ex    (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViPUInt8 val8);
ViStatus _VI_FUNC viOut8Ex   (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViUInt8 val8);
ViStatus _VI_FUNC viIn16Ex   (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViPUInt16 val16);
ViStatus _VI_FUNC viOut16Ex  (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViUInt16 val16);
ViStatus _VI_FUNC viIn32Ex   (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViPUInt32 val32);
ViStatus _VI_FUNC viOut32Ex  (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViUInt32 val32);
ViStatus _VI_FUNC viIn64Ex   (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViPUInt64 val64);
ViStatus _VI_FUNC viOut64Ex  (ViSession vi, ViUInt16 space,
                             ViBusAddress64 offset, ViUInt64 val64);
#endif

ViStatus _VI_FUNC viMoveIn8  (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt8 buf8);
ViStatus _VI_FUNC viMoveOut8 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt8 buf8);
ViStatus _VI_FUNC viMoveIn16 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt16 buf16);
ViStatus _VI_FUNC viMoveOut16 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt16 buf16);
ViStatus _VI_FUNC viMoveIn32 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt32 buf32);
ViStatus _VI_FUNC viMoveOut32 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt32 buf32);

#if defined(_VI_INT64_UINT64_DEFINED)
ViStatus _VI_FUNC viMoveIn64 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt64 buf64);
ViStatus _VI_FUNC viMoveOut64 (ViSession vi, ViUInt16 space, ViBusAddress offset,
                             ViBusSize length, ViAUInt64 buf64);

ViStatus _VI_FUNC viMoveIn8Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt8 buf8);
ViStatus _VI_FUNC viMoveOut8Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt8 buf8);
ViStatus _VI_FUNC viMoveIn16Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt16 buf16);
ViStatus _VI_FUNC viMoveOut16Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt16 buf16);
ViStatus _VI_FUNC viMoveIn32Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt32 buf32);
ViStatus _VI_FUNC viMoveOut32Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt32 buf32);
ViStatus _VI_FUNC viMoveIn64Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt64 buf64);
ViStatus _VI_FUNC viMoveOut64Ex (ViSession vi, ViUInt16 space, ViBusAddress64 offset,
                             ViBusSize length, ViAUInt64 buf64);
#endif

ViStatus _VI_FUNC viMove      (ViSession vi, ViUInt16 srcSpace, ViBusAddress srcOffset,

```

```

ViStatus _VI_FUNC viMoveAsync (ViUInt16 srcWidth, ViUInt16 destSpace,
ViBusAddress destOffset, ViUInt16 destWidth,
ViBusSize srcLength);
ViStatus _VI_FUNC viMoveEx (ViSession vi, ViUInt16 srcSpace, ViBusAddress64 srcOffset,
ViUInt16 srcWidth, ViUInt16 destSpace,
ViBusAddress64 destOffset, ViUInt16 destWidth,
ViBusSize srcLength);
ViStatus _VI_FUNC viMoveAsyncEx (ViSession vi, ViUInt16 srcSpace, ViBusAddress64 srcOffset,
ViUInt16 srcWidth, ViUInt16 destSpace,
ViBusAddress64 destOffset, ViUInt16 destWidth,
ViBusSize srcLength, ViPJobId jobId);

#endif

ViStatus _VI_FUNC viMapAddress (ViSession vi, ViUInt16 mapSpace, ViBusAddress mapOffset,
ViBusSize mapSize, ViBoolean access,
ViAddr suggested, ViPAddr address);
ViStatus _VI_FUNC viUnmapAddress (ViSession vi);

#if defined(_VI_INT64_UINT64_DEFINED)
ViStatus _VI_FUNC viMapAddressEx (ViSession vi, ViUInt16 mapSpace, ViBusAddress64 mapOffset,
ViBusSize mapSize, ViBoolean access,
ViAddr suggested, ViPAddr address);
#endif

void _VI_FUNC viPeek8 (ViSession vi, ViAddr address, ViPUInt8 val8);
void _VI_FUNC viPoke8 (ViSession vi, ViAddr address, ViUInt8 val8);
void _VI_FUNC viPeek16 (ViSession vi, ViAddr address, ViPUInt16 val16);
void _VI_FUNC viPoke16 (ViSession vi, ViAddr address, ViUInt16 val16);
void _VI_FUNC viPeek32 (ViSession vi, ViAddr address, ViPUInt32 val32);
void _VI_FUNC viPoke32 (ViSession vi, ViAddr address, ViUInt32 val32);

#if defined(_VI_INT64_UINT64_DEFINED)
void _VI_FUNC viPeek64 (ViSession vi, ViAddr address, ViPUInt64 val64);
void _VI_FUNC viPoke64 (ViSession vi, ViAddr address, ViUInt64 val64);
#endif

/*- Shared Memory Operations -----*/

ViStatus _VI_FUNC viMemAlloc (ViSession vi, ViBusSize size, ViPBusAddress offset);
ViStatus _VI_FUNC viMemFree (ViSession vi, ViBusAddress offset);

#if defined(_VI_INT64_UINT64_DEFINED)
ViStatus _VI_FUNC viMemAllocEx (ViSession vi, ViBusSize size, ViPBusAddress64 offset);
ViStatus _VI_FUNC viMemFreeEx (ViSession vi, ViBusAddress64 offset);
#endif

/*- Interface Specific Operations -----*/

ViStatus _VI_FUNC viGpibControlREN(ViSession vi, ViUInt16 mode);
ViStatus _VI_FUNC viGpibControlATN(ViSession vi, ViUInt16 mode);
ViStatus _VI_FUNC viGpibSendIFC (ViSession vi);
ViStatus _VI_FUNC viGpibCommand (ViSession vi, ViBuf cmd, ViUInt32 cnt, ViPUInt32 retCnt);
ViStatus _VI_FUNC viGpibPassControl(ViSession vi, ViUInt16 primAddr, ViUInt16 secAddr);

ViStatus _VI_FUNC viVxiCommandQuery(ViSession vi, ViUInt16 mode, ViUInt32 cmd,
ViPUInt32 response);
ViStatus _VI_FUNC viAssertUtilSignal(ViSession vi, ViUInt16 line);
ViStatus _VI_FUNC viAssertIntrSignal(ViSession vi, ViInt16 mode, ViUInt32 statusID);
ViStatus _VI_FUNC viMapTrigger (ViSession vi, ViInt16 trigSrc, ViInt16 trigDest,
ViUInt16 mode);
ViStatus _VI_FUNC viUnmapTrigger (ViSession vi, ViInt16 trigSrc, ViInt16 trigDest);
ViStatus _VI_FUNC viUsbControlOut (ViSession vi, ViInt16 bmRequestType, ViInt16 bRequest,
ViUInt16 wValue, ViUInt16 wIndex, ViUInt16 wLength,
ViBuf buf);
ViStatus _VI_FUNC viUsbControlIn (ViSession vi, ViInt16 bmRequestType, ViInt16 bRequest,
ViUInt16 wValue, ViUInt16 wIndex, ViUInt16 wLength,
ViPBuf buf, ViPUInt16 retCnt);

/*- Attributes (platform independent size) -----*/

```

```

#define VI_ATTR_RSRC_CLASS          (0xBFFF0001UL)
#define VI_ATTR_RSRC_NAME          (0xBFFF0002UL)
#define VI_ATTR_RSRC_IMPL_VERSION (0x3FFF0003UL)
#define VI_ATTR_RSRC_LOCK_STATE   (0x3FFF0004UL)
#define VI_ATTR_MAX_QUEUE_LENGTH  (0x3FFF0005UL)
#define VI_ATTR_USER_DATA_32      (0x3FFF0007UL)
#define VI_ATTR_FDC_CHNL          (0x3FFF000DUL)
#define VI_ATTR_FDC_MODE          (0x3FFF000FUL)
#define VI_ATTR_FDC_GEN_SIGNAL_EN (0x3FFF0011UL)
#define VI_ATTR_FDC_USE_PAIR      (0x3FFF0013UL)
#define VI_ATTR_SEND_END_EN       (0x3FFF0016UL)
#define VI_ATTR_TERMCHAR          (0x3FFF0018UL)
#define VI_ATTR_TMO_VALUE         (0x3FFF001AUL)
#define VI_ATTR_GPIB_READDR_EN    (0x3FFF001BUL)
#define VI_ATTR_IO_PROT           (0x3FFF001CUL)
#define VI_ATTR_DMA_ALLOW_EN      (0x3FFF001EUL)
#define VI_ATTR_ASRL_BAUD         (0x3FFF0021UL)
#define VI_ATTR_ASRL_DATA_BITS   (0x3FFF0022UL)
#define VI_ATTR_ASRL_PARITY       (0x3FFF0023UL)
#define VI_ATTR_ASRL_STOP_BITS   (0x3FFF0024UL)
#define VI_ATTR_ASRL_FLOW_CNTRL  (0x3FFF0025UL)
#define VI_ATTR_RD_BUF_OPER_MODE  (0x3FFF002AUL)
#define VI_ATTR_RD_BUF_SIZE      (0x3FFF002BUL)
#define VI_ATTR_WR_BUF_OPER_MODE  (0x3FFF002DUL)
#define VI_ATTR_WR_BUF_SIZE      (0x3FFF002EUL)
#define VI_ATTR_SUPPRESS_END_EN   (0x3FFF0036UL)
#define VI_ATTR_TERMCHAR_EN      (0x3FFF0038UL)
#define VI_ATTR_DEST_ACCESS_PRIV  (0x3FFF0039UL)
#define VI_ATTR_DEST_BYTE_ORDER  (0x3FFF003AUL)
#define VI_ATTR_SRC_ACCESS_PRIV   (0x3FFF003CUL)
#define VI_ATTR_SRC_BYTE_ORDER   (0x3FFF003DUL)
#define VI_ATTR_SRC_INCREMENT     (0x3FFF0040UL)
#define VI_ATTR_DEST_INCREMENT    (0x3FFF0041UL)
#define VI_ATTR_WIN_ACCESS_PRIV   (0x3FFF0045UL)
#define VI_ATTR_WIN_BYTE_ORDER   (0x3FFF0047UL)
#define VI_ATTR_GPIB_ATN_STATE    (0x3FFF0057UL)
#define VI_ATTR_GPIB_ADDR_STATE   (0x3FFF005CUL)
#define VI_ATTR_GPIB_CIC_STATE    (0x3FFF005EUL)
#define VI_ATTR_GPIB_NDAC_STATE   (0x3FFF0062UL)
#define VI_ATTR_GPIB_SRQ_STATE    (0x3FFF0067UL)
#define VI_ATTR_GPIB_SYS_CNTRL_STATE (0x3FFF0068UL)
#define VI_ATTR_GPIB_HS488_CBL_LEN (0x3FFF0069UL)
#define VI_ATTR_CMDR_LA           (0x3FFF006BUL)
#define VI_ATTR_VXI_DEV_CLASS     (0x3FFF006CUL)
#define VI_ATTR_MAINFRAME_LA      (0x3FFF0070UL)
#define VI_ATTR_MANF_NAME         (0xBFFF0072UL)
#define VI_ATTR_MODEL_NAME        (0xBFFF0077UL)
#define VI_ATTR_VXI_VME_INTR_STATUS (0x3FFF008BUL)
#define VI_ATTR_VXI_VME_TRIG_STATUS (0x3FFF008DUL)
#define VI_ATTR_VXI_VME_SYSFAIL_STATE (0x3FFF0094UL)
#define VI_ATTR_WIN_BASE_ADDR_32  (0x3FFF0098UL)
#define VI_ATTR_WIN_SIZE_32       (0x3FFF009AUL)
#define VI_ATTR_ASRL_AVAIL_NUM    (0x3FFF00ACUL)
#define VI_ATTR_MEM_BASE_32       (0x3FFF00ADUL)
#define VI_ATTR_ASRL_CTS_STATE    (0x3FFF00AEUL)
#define VI_ATTR_ASRL_DCD_STATE    (0x3FFF00AFUL)
#define VI_ATTR_ASRL_DSR_STATE    (0x3FFF00B1UL)
#define VI_ATTR_ASRL_DTR_STATE    (0x3FFF00B2UL)
#define VI_ATTR_ASRL_END_IN       (0x3FFF00B3UL)
#define VI_ATTR_ASRL_END_OUT      (0x3FFF00B4UL)
#define VI_ATTR_ASRL_REPLACE_CHAR (0x3FFF00BEUL)
#define VI_ATTR_ASRL_RI_STATE     (0x3FFF00BFUL)
#define VI_ATTR_ASRL_RTS_STATE    (0x3FFF00C0UL)
#define VI_ATTR_ASRL_XON_CHAR     (0x3FFF00C1UL)
#define VI_ATTR_ASRL_XOFF_CHAR    (0x3FFF00C2UL)
#define VI_ATTR_WIN_ACCESS        (0x3FFF00C3UL)
#define VI_ATTR_RM_SESSION        (0x3FFF00C4UL)
#define VI_ATTR_VXI_LA            (0x3FFF00D5UL)
#define VI_ATTR_MANF_ID           (0x3FFF00D9UL)
#define VI_ATTR_MEM_SIZE_32       (0x3FFF00DDUL)
#define VI_ATTR_MEM_SPACE         (0x3FFF00DEUL)
#define VI_ATTR_MODEL_CODE        (0x3FFF00DFUL)
#define VI_ATTR_SLOT              (0x3FFF00E8UL)
#define VI_ATTR_INTF_INST_NAME    (0xBFFF00E9UL)
#define VI_ATTR_IMMEDIATE_SERV    (0x3FFF0100UL)
#define VI_ATTR_INTF_PARENT_NUM   (0x3FFF0101UL)
#define VI_ATTR_RSRC_SPEC_VERSION (0x3FFF0170UL)

```

```

#define VI_ATTR_INTF_TYPE (0x3FFF0171UL)
#define VI_ATTR_GPIB_PRIMARY_ADDR (0x3FFF0172UL)
#define VI_ATTR_GPIB_SECONDARY_ADDR (0x3FFF0173UL)
#define VI_ATTR_RSRC_MANF_NAME (0xBFFF0174UL)
#define VI_ATTR_RSRC_MANF_ID (0x3FFF0175UL)
#define VI_ATTR_INTF_NUM (0x3FFF0176UL)
#define VI_ATTR_TRIG_ID (0x3FFF0177UL)
#define VI_ATTR_GPIB_REN_STATE (0x3FFF0181UL)
#define VI_ATTR_GPIB_UNADDR_EN (0x3FFF0184UL)
#define VI_ATTR_DEV_STATUS_BYTE (0x3FFF0189UL)
#define VI_ATTR_FILE_APPEND_EN (0x3FFF0192UL)
#define VI_ATTR_VXI_TRIG_SUPPORT (0x3FFF0194UL)
#define VI_ATTR_TCPIP_ADDR (0xBFFF0195UL)
#define VI_ATTR_TCPIP_HOSTNAME (0xBFFF0196UL)
#define VI_ATTR_TCPIP_PORT (0x3FFF0197UL)
#define VI_ATTR_TCPIP_DEVICE_NAME (0xBFFF0199UL)
#define VI_ATTR_TCPIP_NODELAY (0x3FFF019AUL)
#define VI_ATTR_TCPIP_KEEPLIVE (0x3FFF019BUL)
#define VI_ATTR_4882_COMPLIANT (0x3FFF019FUL)
#define VI_ATTR_USB_SERIAL_NUM (0xBFFF01A0UL)
#define VI_ATTR_USB_INTFC_NUM (0x3FFF01A1UL)
#define VI_ATTR_USB_PROTOCOL (0x3FFF01A7UL)
#define VI_ATTR_USB_MAX_INTR_SIZE (0x3FFF01AFUL)
#define VI_ATTR_PXI_DEV_NUM (0x3FFF0201UL)
#define VI_ATTR_PXI_FUNC_NUM (0x3FFF0202UL)
#define VI_ATTR_PXI_BUS_NUM (0x3FFF0205UL)
#define VI_ATTR_PXI_CHASSIS (0x3FFF0206UL)
#define VI_ATTR_PXI_SLOTPATH (0xBFFF0207UL)
#define VI_ATTR_PXI_SLOT_LBUS_LEFT (0x3FFF0208UL)
#define VI_ATTR_PXI_SLOT_LBUS_RIGHT (0x3FFF0209UL)
#define VI_ATTR_PXI_TRIG_BUS (0x3FFF020AUL)
#define VI_ATTR_PXI_STAR_TRIG_BUS (0x3FFF020BUL)
#define VI_ATTR_PXI_STAR_TRIG_LINE (0x3FFF020CUL)
#define VI_ATTR_PXI_MEM_TYPE_BAR0 (0x3FFF0211UL)
#define VI_ATTR_PXI_MEM_TYPE_BAR1 (0x3FFF0212UL)
#define VI_ATTR_PXI_MEM_TYPE_BAR2 (0x3FFF0213UL)
#define VI_ATTR_PXI_MEM_TYPE_BAR3 (0x3FFF0214UL)
#define VI_ATTR_PXI_MEM_TYPE_BAR4 (0x3FFF0215UL)
#define VI_ATTR_PXI_MEM_TYPE_BAR5 (0x3FFF0216UL)
#define VI_ATTR_PXI_MEM_BASE_BAR0 (0x3FFF0221UL)
#define VI_ATTR_PXI_MEM_BASE_BAR1 (0x3FFF0222UL)
#define VI_ATTR_PXI_MEM_BASE_BAR2 (0x3FFF0223UL)
#define VI_ATTR_PXI_MEM_BASE_BAR3 (0x3FFF0224UL)
#define VI_ATTR_PXI_MEM_BASE_BAR4 (0x3FFF0225UL)
#define VI_ATTR_PXI_MEM_BASE_BAR5 (0x3FFF0226UL)
#define VI_ATTR_PXI_MEM_SIZE_BAR0 (0x3FFF0231UL)
#define VI_ATTR_PXI_MEM_SIZE_BAR1 (0x3FFF0232UL)
#define VI_ATTR_PXI_MEM_SIZE_BAR2 (0x3FFF0233UL)
#define VI_ATTR_PXI_MEM_SIZE_BAR3 (0x3FFF0234UL)
#define VI_ATTR_PXI_MEM_SIZE_BAR4 (0x3FFF0235UL)
#define VI_ATTR_PXI_MEM_SIZE_BAR5 (0x3FFF0236UL)
#define VI_ATTR_PXI_IS_EXPRESS (0x3FFF0240UL)
#define VI_ATTR_PXI_SLOT_LWIDTH (0x3FFF0241UL)
#define VI_ATTR_PXI_MAX_LWIDTH (0x3FFF0242UL)
#define VI_ATTR_PXI_ACTUAL_LWIDTH (0x3FFF0243UL)
#define VI_ATTR_PXI_DSTAR_BUS (0x3FFF0244UL)
#define VI_ATTR_PXI_DSTAR_SET (0x3FFF0245UL)

#define VI_ATTR_TCPIP_HISLIP_OVERLAP_EN (0x3FFF0300UL)
#define VI_ATTR_TCPIP_HISLIP_VERSION (0x3FFF0301UL)
#define VI_ATTR_TCPIP_HISLIP_MAX_MESSAGE_KB (0x3FFF0302UL)

#define VI_ATTR_JOB_ID (0x3FFF4006UL)
#define VI_ATTR_EVENT_TYPE (0x3FFF4010UL)
#define VI_ATTR_SIGP_STATUS_ID (0x3FFF4011UL)
#define VI_ATTR_RECV_TRIG_ID (0x3FFF4012UL)
#define VI_ATTR_INTR_STATUS_ID (0x3FFF4023UL)
#define VI_ATTR_STATUS (0x3FFF4025UL)
#define VI_ATTR_RET_COUNT_32 (0x3FFF4026UL)
#define VI_ATTR_BUFFER (0x3FFF4027UL)
#define VI_ATTR_RECV_INTR_LEVEL (0x3FFF4041UL)
#define VI_ATTR_OPER_NAME (0xBFFF4042UL)
#define VI_ATTR_GPIB_RECV_CIC_STATE (0x3FFF4193UL)
#define VI_ATTR_RECV_TCPIP_ADDR (0xBFFF4198UL)
#define VI_ATTR_USB_RECV_INTR_SIZE (0x3FFF41B0UL)
#define VI_ATTR_USB_RECV_INTR_DATA (0xBFFF41B1UL)

```

```

/*- Attributes (platform dependent size) -----*/

#if defined(_VI_INT64_UINT64_DEFINED) && defined(_VISA_ENV_IS_64_BIT)
#define VI_ATTR_USER_DATA_64      (0x3FFF000AUL)
#define VI_ATTR_RET_COUNT_64     (0x3FFF4028UL)
#define VI_ATTR_USER_DATA        (VI_ATTR_USER_DATA_64)
#define VI_ATTR_RET_COUNT       (VI_ATTR_RET_COUNT_64)
#else
#define VI_ATTR_USER_DATA        (VI_ATTR_USER_DATA_32)
#define VI_ATTR_RET_COUNT       (VI_ATTR_RET_COUNT_32)
#endif

#if defined(_VI_INT64_UINT64_DEFINED)
#define VI_ATTR_WIN_BASE_ADDR_64  (0x3FFF009BUL)
#define VI_ATTR_WIN_SIZE_64      (0x3FFF009CUL)
#define VI_ATTR_MEM_BASE_64      (0x3FFF00D0UL)
#define VI_ATTR_MEM_SIZE_64      (0x3FFF00D1UL)
#endif

#if defined(_VI_INT64_UINT64_DEFINED) && defined(_VISA_ENV_IS_64_BIT)
#define VI_ATTR_WIN_BASE_ADDR     (VI_ATTR_WIN_BASE_ADDR_64)
#define VI_ATTR_WIN_SIZE         (VI_ATTR_WIN_SIZE_64)
#define VI_ATTR_MEM_BASE         (VI_ATTR_MEM_BASE_64)
#define VI_ATTR_MEM_SIZE         (VI_ATTR_MEM_SIZE_64)
#else
#define VI_ATTR_WIN_BASE_ADDR     (VI_ATTR_WIN_BASE_ADDR_32)
#define VI_ATTR_WIN_SIZE         (VI_ATTR_WIN_SIZE_32)
#define VI_ATTR_MEM_BASE         (VI_ATTR_MEM_BASE_32)
#define VI_ATTR_MEM_SIZE         (VI_ATTR_MEM_SIZE_32)
#endif

/*- Event Types -----*/

#define VI_EVENT_IO_COMPLETION    (0x3FFF2009UL)
#define VI_EVENT_TRIG            (0xBFFF200AUL)
#define VI_EVENT_SERVICE_REQ     (0x3FFF200BUL)
#define VI_EVENT_CLEAR           (0x3FFF200DUL)
#define VI_EVENT_EXCEPTION       (0xBFFF200EUL)
#define VI_EVENT_GPIB_CIC       (0x3FFF2012UL)
#define VI_EVENT_GPIB_TALK       (0x3FFF2013UL)
#define VI_EVENT_GPIB_LISTEN    (0x3FFF2014UL)
#define VI_EVENT_VXI_VME_SYSFAIL (0x3FFF201DUL)
#define VI_EVENT_VXI_VME_SYSRESET (0x3FFF201EUL)
#define VI_EVENT_VXI_SIGP        (0x3FFF2020UL)
#define VI_EVENT_VXI_VME_INTR   (0xBFFF2021UL)
#define VI_EVENT_PXI_INTR        (0x3FFF2022UL)
#define VI_EVENT_TCPIP_CONNECT  (0x3FFF2036UL)
#define VI_EVENT_USB_INTR        (0x3FFF2037UL)

#define VI_ALL_ENABLED_EVENTS    (0x3FFF7FFFUL)

/*- Completion and Error Codes -----*/

#define VI_SUCCESS_EVENT_EN      (0x3FFF0002L) /* 3FFF0002, 1073676290 */
#define VI_SUCCESS_EVENT_DIS     (0x3FFF0003L) /* 3FFF0003, 1073676291 */
#define VI_SUCCESS_QUEUE_EMPTY  (0x3FFF0004L) /* 3FFF0004, 1073676292 */
#define VI_SUCCESS_TERM_CHAR    (0x3FFF0005L) /* 3FFF0005, 1073676293 */
#define VI_SUCCESS_MAX_CNT      (0x3FFF0006L) /* 3FFF0006, 1073676294 */
#define VI_SUCCESS_DEV_NPRESENT (0x3FFF0007DL) /* 3FFF007D, 1073676413 */
#define VI_SUCCESS_TRIG_MAPPED  (0x3FFF0007EL) /* 3FFF007E, 1073676414 */
#define VI_SUCCESS_QUEUE_NEMPTY (0x3FFF0080L) /* 3FFF0080, 1073676416 */
#define VI_SUCCESS_NCHAIN       (0x3FFF0098L) /* 3FFF0098, 1073676440 */
#define VI_SUCCESS_NESTED_SHARED (0x3FFF0099L) /* 3FFF0099, 1073676441 */
#define VI_SUCCESS_NESTED_EXCLUSIVE (0x3FFF009AL) /* 3FFF009A, 1073676442 */
#define VI_SUCCESS_SYNC         (0x3FFF009BL) /* 3FFF009B, 1073676443 */

#define VI_WARN_QUEUE_OVERFLOW   (0x3FFF000CL) /* 3FFF000C, 1073676300 */
#define VI_WARN_CONFIG_NLOADED  (0x3FFF0007L) /* 3FFF0077, 1073676407 */
#define VI_WARN_NULL_OBJECT     (0x3FFF0082L) /* 3FFF0082, 1073676418 */
#define VI_WARN_NSUP_ATTR_STATE (0x3FFF0084L) /* 3FFF0084, 1073676420 */
#define VI_WARN_UNKNOWN_STATUS  (0x3FFF0085L) /* 3FFF0085, 1073676421 */
#define VI_WARN_UNKNUP_BUF      (0x3FFF0088L) /* 3FFF0088, 1073676424 */
#define VI_WARN_EXT_FUNC_NIMPL  (0x3FFF00A9L) /* 3FFF00A9, 1073676457 */

#define VI_ERROR_SYSTEM_ERROR   (_VI_ERROR+0x3FFF0000L) /* BFFF0000, -1073807360 */
#define VI_ERROR_INV_OBJECT     (_VI_ERROR+0x3FFF000EL) /* BFFF000E, -1073807346 */
#define VI_ERROR_RSRC_LOCKED    (_VI_ERROR+0x3FFF000FL) /* BFFF000F, -1073807345 */
#define VI_ERROR_INV_EXPR       (_VI_ERROR+0x3FFF0010L) /* BFFF0010, -1073807344 */
#define VI_ERROR_RSRC_NFOUND     (_VI_ERROR+0x3FFF0011L) /* BFFF0011, -1073807343 */

```

```

#define VI_ERROR_INV_RSRC_NAME      (_VI_ERROR+0x3FFF0012L) /* BFFF0012, -1073807342 */
#define VI_ERROR_INV_ACC_MODE      (_VI_ERROR+0x3FFF0013L) /* BFFF0013, -1073807341 */
#define VI_ERROR_TMO                (_VI_ERROR+0x3FFF0015L) /* BFFF0015, -1073807339 */
#define VI_ERROR_CLOSING_FAILED     (_VI_ERROR+0x3FFF0016L) /* BFFF0016, -1073807338 */
#define VI_ERROR_INV_DEGREE        (_VI_ERROR+0x3FFF001BL) /* BFFF001B, -1073807333 */
#define VI_ERROR_INV_JOB_ID        (_VI_ERROR+0x3FFF001CL) /* BFFF001C, -1073807332 */
#define VI_ERROR_NSUP_ATTR         (_VI_ERROR+0x3FFF001DL) /* BFFF001D, -1073807331 */
#define VI_ERROR_NSUP_ATTR_STATE   (_VI_ERROR+0x3FFF001EL) /* BFFF001E, -1073807330 */
#define VI_ERROR_ATTR_READONLY     (_VI_ERROR+0x3FFF001FL) /* BFFF001F, -1073807329 */
#define VI_ERROR_INV_LOCK_TYPE     (_VI_ERROR+0x3FFF0020L) /* BFFF0020, -1073807328 */
#define VI_ERROR_INV_ACCESS_KEY    (_VI_ERROR+0x3FFF0021L) /* BFFF0021, -1073807327 */
#define VI_ERROR_INV_EVENT         (_VI_ERROR+0x3FFF0026L) /* BFFF0026, -1073807322 */
#define VI_ERROR_INV_MECH          (_VI_ERROR+0x3FFF0027L) /* BFFF0027, -1073807321 */
#define VI_ERROR_HNDLR_NINSTALLED  (_VI_ERROR+0x3FFF0028L) /* BFFF0028, -1073807320 */
#define VI_ERROR_INV_HNDLR_REF     (_VI_ERROR+0x3FFF0029L) /* BFFF0029, -1073807319 */
#define VI_ERROR_INV_CONTEXT       (_VI_ERROR+0x3FFF002AL) /* BFFF002A, -1073807318 */
#define VI_ERROR_NENABLED         (_VI_ERROR+0x3FFF002FL) /* BFFF002F, -1073807313 */
#define VI_ERROR_ABORT             (_VI_ERROR+0x3FFF0030L) /* BFFF0030, -1073807312 */
#define VI_ERROR_RAW_WR_PROT_VIOL  (_VI_ERROR+0x3FFF0034L) /* BFFF0034, -1073807308 */
#define VI_ERROR_RAW_RD_PROT_VIOL  (_VI_ERROR+0x3FFF0035L) /* BFFF0035, -1073807307 */
#define VI_ERROR_OUTP_PROT_VIOL    (_VI_ERROR+0x3FFF0036L) /* BFFF0036, -1073807306 */
#define VI_ERROR_INP_PROT_VIOL     (_VI_ERROR+0x3FFF0037L) /* BFFF0037, -1073807305 */
#define VI_ERROR_BERR              (_VI_ERROR+0x3FFF0038L) /* BFFF0038, -1073807304 */
#define VI_ERROR_IN_PROGRESS       (_VI_ERROR+0x3FFF0039L) /* BFFF0039, -1073807303 */
#define VI_ERROR_INV_SETUP         (_VI_ERROR+0x3FFF003AL) /* BFFF003A, -1073807302 */
#define VI_ERROR_QUEUE_ERROR      (_VI_ERROR+0x3FFF003BL) /* BFFF003B, -1073807301 */
#define VI_ERROR_ALLOC             (_VI_ERROR+0x3FFF003CL) /* BFFF003C, -1073807300 */
#define VI_ERROR_INV_MASK         (_VI_ERROR+0x3FFF003DL) /* BFFF003D, -1073807299 */
#define VI_ERROR_IO                (_VI_ERROR+0x3FFF003EL) /* BFFF003E, -1073807298 */
#define VI_ERROR_INV_FMT           (_VI_ERROR+0x3FFF003FL) /* BFFF003F, -1073807297 */
#define VI_ERROR_NSUP_FMT         (_VI_ERROR+0x3FFF0041L) /* BFFF0041, -1073807295 */
#define VI_ERROR_LINE_IN_USE       (_VI_ERROR+0x3FFF0042L) /* BFFF0042, -1073807294 */
#define VI_ERROR_NSUP_MODE         (_VI_ERROR+0x3FFF0046L) /* BFFF0046, -1073807290 */
#define VI_ERROR_SRQ_NOCCURRED     (_VI_ERROR+0x3FFF004AL) /* BFFF004A, -1073807286 */
#define VI_ERROR_INV_SPACE        (_VI_ERROR+0x3FFF004EL) /* BFFF004E, -1073807282 */
#define VI_ERROR_INV_OFFSET       (_VI_ERROR+0x3FFF0051L) /* BFFF0051, -1073807279 */
#define VI_ERROR_INV_WIDTH        (_VI_ERROR+0x3FFF0052L) /* BFFF0052, -1073807278 */
#define VI_ERROR_NSUP_OFFSET      (_VI_ERROR+0x3FFF0054L) /* BFFF0054, -1073807276 */
#define VI_ERROR_NSUP_VAR_WIDTH   (_VI_ERROR+0x3FFF0055L) /* BFFF0055, -1073807275 */
#define VI_ERROR_WINDOW_NMAPPED   (_VI_ERROR+0x3FFF0057L) /* BFFF0057, -1073807273 */
#define VI_ERROR_RESP_PENDING     (_VI_ERROR+0x3FFF0059L) /* BFFF0059, -1073807271 */
#define VI_ERROR_NLISTENERS       (_VI_ERROR+0x3FFF005FL) /* BFFF005F, -1073807265 */
#define VI_ERROR_NCIC             (_VI_ERROR+0x3FFF0060L) /* BFFF0060, -1073807264 */
#define VI_ERROR_NSYS_CNTLRL      (_VI_ERROR+0x3FFF0061L) /* BFFF0061, -1073807263 */
#define VI_ERROR_NSUP_OPER        (_VI_ERROR+0x3FFF0067L) /* BFFF0067, -1073807257 */
#define VI_ERROR_INTR_PENDING     (_VI_ERROR+0x3FFF0068L) /* BFFF0068, -1073807256 */
#define VI_ERROR_ASRL_PARITY      (_VI_ERROR+0x3FFF006AL) /* BFFF006A, -1073807254 */
#define VI_ERROR_ASRL_FRAMING     (_VI_ERROR+0x3FFF006BL) /* BFFF006B, -1073807253 */
#define VI_ERROR_ASRL_OVERRUN     (_VI_ERROR+0x3FFF006CL) /* BFFF006C, -1073807252 */
#define VI_ERROR_TRIG_NMAPPED     (_VI_ERROR+0x3FFF006EL) /* BFFF006E, -1073807250 */
#define VI_ERROR_NSUP_ALIGN_OFFSET (_VI_ERROR+0x3FFF0070L) /* BFFF0070, -1073807248 */
#define VI_ERROR_USER_BUF         (_VI_ERROR+0x3FFF0071L) /* BFFF0071, -1073807247 */
#define VI_ERROR_RSRC_BUSY        (_VI_ERROR+0x3FFF0072L) /* BFFF0072, -1073807246 */
#define VI_ERROR_NSUP_WIDTH       (_VI_ERROR+0x3FFF0076L) /* BFFF0076, -1073807242 */
#define VI_ERROR_INV_PARAMETER    (_VI_ERROR+0x3FFF0078L) /* BFFF0078, -1073807240 */
#define VI_ERROR_INV_PROT         (_VI_ERROR+0x3FFF0079L) /* BFFF0079, -1073807239 */
#define VI_ERROR_INV_SIZE         (_VI_ERROR+0x3FFF007BL) /* BFFF007B, -1073807237 */
#define VI_ERROR_WINDOW_MAPPED   (_VI_ERROR+0x3FFF0080L) /* BFFF0080, -1073807232 */
#define VI_ERROR_NIMPL_OPER       (_VI_ERROR+0x3FFF0081L) /* BFFF0081, -1073807231 */
#define VI_ERROR_INV_LENGTH       (_VI_ERROR+0x3FFF0083L) /* BFFF0083, -1073807229 */
#define VI_ERROR_INV_MODE         (_VI_ERROR+0x3FFF0091L) /* BFFF0091, -1073807215 */
#define VI_ERROR_SESN_NLOCKED     (_VI_ERROR+0x3FFF009CL) /* BFFF009C, -1073807204 */
#define VI_ERROR_MEM_NSHARED      (_VI_ERROR+0x3FFF009DL) /* BFFF009D, -1073807203 */
#define VI_ERROR_LIBRARY_NFOUND   (_VI_ERROR+0x3FFF009EL) /* BFFF009E, -1073807202 */
#define VI_ERROR_NSUP_INTR        (_VI_ERROR+0x3FFF009FL) /* BFFF009F, -1073807201 */
#define VI_ERROR_INV_LINE         (_VI_ERROR+0x3FFF00A0L) /* BFFF00A0, -1073807200 */
#define VI_ERROR_FILE_ACCESS      (_VI_ERROR+0x3FFF00A1L) /* BFFF00A1, -1073807199 */
#define VI_ERROR_FILE_IO          (_VI_ERROR+0x3FFF00A2L) /* BFFF00A2, -1073807198 */
#define VI_ERROR_NSUP_LINE        (_VI_ERROR+0x3FFF00A3L) /* BFFF00A3, -1073807197 */
#define VI_ERROR_NSUP_MECH        (_VI_ERROR+0x3FFF00A4L) /* BFFF00A4, -1073807196 */
#define VI_ERROR_INTF_NUM_NCONFIG (_VI_ERROR+0x3FFF00A5L) /* BFFF00A5, -1073807195 */
#define VI_ERROR_CONN_LOST        (_VI_ERROR+0x3FFF00A6L) /* BFFF00A6, -1073807194 */
#define VI_ERROR_NPERMISSION      (_VI_ERROR+0x3FFF00A8L) /* BFFF00A8, -1073807192 */

/*- Other VISA Definitions -----*/

#define VI_VERSION_MAJOR(ver)      (((ViVersion)ver) & 0xFFFF0000UL) >> 20)

```

```

#define VI_VERSION_MINOR(ver)      (((ViVersion)ver) & 0x000FFF00UL) >> 8)
#define VI_VERSION_SUBMINOR(ver)  (((ViVersion)ver) & 0x000000FFUL)

#define VI_FIND_BUFLen           (256)

#define VI_INTF_GPIB              (1)
#define VI_INTF_VXI               (2)
#define VI_INTF_GPIB_VXI         (3)
#define VI_INTF_ASRL              (4)
#define VI_INTF_PXI               (5)
#define VI_INTF_TCPIP             (6)
#define VI_INTF_USB               (7)

#define VI_PROT_NORMAL            (1)
#define VI_PROT_FDC               (2)
#define VI_PROT_HS488             (3)
#define VI_PROT_4882_STRS        (4)
#define VI_PROT_USBTMC_VENDOR    (5)

#define VI_FDC_NORMAL            (1)
#define VI_FDC_STREAM             (2)

#define VI_LOCAL_SPACE            (0)
#define VI_A16_SPACE              (1)
#define VI_A24_SPACE              (2)
#define VI_A32_SPACE              (3)
#define VI_A64_SPACE              (4)
#define VI_PXI_ALLOC_SPACE        (9)
#define VI_PXI_CFG_SPACE          (10)
#define VI_PXI_BAR0_SPACE         (11)
#define VI_PXI_BAR1_SPACE         (12)
#define VI_PXI_BAR2_SPACE         (13)
#define VI_PXI_BAR3_SPACE         (14)
#define VI_PXI_BAR4_SPACE         (15)
#define VI_PXI_BAR5_SPACE         (16)
#define VI_OPAQUE_SPACE           (0xFFFF)

#define VI_UNKNOWN_LA             (-1)
#define VI_UNKNOWN_SLOT           (-1)
#define VI_UNKNOWN_LEVEL          (-1)
#define VI_UNKNOWN_CHASSIS        (-1)

#define VI_QUEUE                  (1)
#define VI_HNDLR                  (2)
#define VI_SUSPEND_HNDLR          (4)
#define VI_ALL_MECH                (0xFFFF)

#define VI_ANY_HNDLR              (0)

#define VI_TRIG_ALL                (-2)
#define VI_TRIG_SW                 (-1)
#define VI_TRIG_TTL0               (0)
#define VI_TRIG_TTL1               (1)
#define VI_TRIG_TTL2               (2)
#define VI_TRIG_TTL3               (3)
#define VI_TRIG_TTL4               (4)
#define VI_TRIG_TTL5               (5)
#define VI_TRIG_TTL6               (6)
#define VI_TRIG_TTL7               (7)
#define VI_TRIG_ECL0               (8)
#define VI_TRIG_ECL1               (9)
#define VI_TRIG_PANEL_IN           (27)
#define VI_TRIG_PANEL_OUT          (28)

#define VI_TRIG_PROT_DEFAULT       (0)
#define VI_TRIG_PROT_ON            (1)
#define VI_TRIG_PROT_OFF           (2)
#define VI_TRIG_PROT_SYNC          (5)
#define VI_TRIG_PROT_RESERVE       (6)
#define VI_TRIG_PROT_UNRESERVE     (7)

#define VI_READ_BUF                (1)
#define VI_WRITE_BUF               (2)
#define VI_READ_BUF_DISCARD        (4)
#define VI_WRITE_BUF_DISCARD       (8)
#define VI_IO_IN_BUF               (16)

```

```

#define VI_IO_OUT_BUF (32)
#define VI_IO_IN_BUF_DISCARD (64)
#define VI_IO_OUT_BUF_DISCARD (128)

#define VI_FLUSH_ON_ACCESS (1)
#define VI_FLUSH_WHEN_FULL (2)
#define VI_FLUSH_DISABLE (3)

#define VI_NMAPPED (1)
#define VI_USE_OPERS (2)
#define VI_DEREF_ADDR (3)

#define VI_TMO_IMMEDIATE (0L)
#define VI_TMO_INFINITE (0xFFFFFFFFFUL)

#define VI_NO_LOCK (0)
#define VI_EXCLUSIVE_LOCK (1)
#define VI_SHARED_LOCK (2)
#define VI_LOAD_CONFIG (4)

#define VI_NO_SEC_ADDR (0xFFFF)

#define VI_ASRL_PAR_NONE (0)
#define VI_ASRL_PAR_ODD (1)
#define VI_ASRL_PAR_EVEN (2)
#define VI_ASRL_PAR_MARK (3)
#define VI_ASRL_PAR_SPACE (4)

#define VI_ASRL_STOP_ONE (10)
#define VI_ASRL_STOP_ONE5 (15)
#define VI_ASRL_STOP_TWO (20)

#define VI_ASRL_FLOW_NONE (0)
#define VI_ASRL_FLOW_XON_XOFF (1)
#define VI_ASRL_FLOW_RTS_CTS (2)
#define VI_ASRL_FLOW_DTR_DSR (4)

#define VI_ASRL_END_NONE (0)
#define VI_ASRL_END_LAST_BIT (1)
#define VI_ASRL_END_TERMCHAR (2)
#define VI_ASRL_END_BREAK (3)

#define VI_STATE_ASSERTED (1)
#define VI_STATE_UNASSERTED (0)
#define VI_STATE_UNKNOWN (-1)

#define VI_BIG_ENDIAN (0)
#define VI_LITTLE_ENDIAN (1)

#define VI_DATA_PRIV (0)
#define VI_DATA_NPRIV (1)
#define VI_PROG_PRIV (2)
#define VI_PROG_NPRIV (3)
#define VI_BLCK_PRIV (4)
#define VI_BLCK_NPRIV (5)
#define VI_D64_PRIV (6)
#define VI_D64_NPRIV (7)

#define VI_WIDTH_8 (1)
#define VI_WIDTH_16 (2)
#define VI_WIDTH_32 (4)
#define VI_WIDTH_64 (8)

#define VI_GPIB_REN_DEASSERT (0)
#define VI_GPIB_REN_ASSERT (1)
#define VI_GPIB_REN_DEASSERT_GTL (2)
#define VI_GPIB_REN_ASSERT_ADDRESS (3)
#define VI_GPIB_REN_ASSERT_LLO (4)
#define VI_GPIB_REN_ASSERT_ADDRESS_LLO (5)
#define VI_GPIB_REN_ADDRESS_GTL (6)

#define VI_GPIB_ATN_DEASSERT (0)
#define VI_GPIB_ATN_ASSERT (1)
#define VI_GPIB_ATN_DEASSERT_HANDSHAKE (2)
#define VI_GPIB_ATN_ASSERT_IMMEDIATE (3)

#define VI_GPIB_HS488_DISABLED (0)

```

```

#define VI_GPIB_HS488_NIMPL          (-1)

#define VI_GPIB_UNADDRESSED          (0)
#define VI_GPIB_TALKER               (1)
#define VI_GPIB_LISTENER             (2)

#define VI_VXI_CMD16                  (0x0200)
#define VI_VXI_CMD16_RESP16          (0x0202)
#define VI_VXI_RESP16                 (0x0002)
#define VI_VXI_CMD32                  (0x0400)
#define VI_VXI_CMD32_RESP32          (0x0402)
#define VI_VXI_CMD32_RESP32          (0x0404)
#define VI_VXI_RESP32                 (0x0004)

#define VI_ASSERT_SIGNAL              (-1)
#define VI_ASSERT_USE_ASSIGNED        (0)
#define VI_ASSERT_IRQ1                (1)
#define VI_ASSERT_IRQ2                (2)
#define VI_ASSERT_IRQ3                (3)
#define VI_ASSERT_IRQ4                (4)
#define VI_ASSERT_IRQ5                (5)
#define VI_ASSERT_IRQ6                (6)
#define VI_ASSERT_IRQ7                (7)

#define VI_UTIL_ASSERT_SYSRESET        (1)
#define VI_UTIL_ASSERT_SYSFAIL        (2)
#define VI_UTIL_DEASSERT_SYSFAIL      (3)

#define VI_VXI_CLASS_MEMORY            (0)
#define VI_VXI_CLASS_EXTENDED         (1)
#define VI_VXI_CLASS_MESSAGE          (2)
#define VI_VXI_CLASS_REGISTER         (3)
#define VI_VXI_CLASS_OTHER            (4)
#define VI_PXI_ADDR_NONE               (0)
#define VI_PXI_ADDR_MEM                (1)
#define VI_PXI_ADDR_IO                 (2)
#define VI_PXI_ADDR_CFG                (3)

#define VI_TRIG_UNKNOWN                (-1)
#define VI_PXI_LBUS_STAR_TRIG_BUS_0    (1000)
#define VI_PXI_LBUS_STAR_TRIG_BUS_1    (1001)
#define VI_PXI_LBUS_STAR_TRIG_BUS_2    (1002)
#define VI_PXI_LBUS_STAR_TRIG_BUS_3    (1003)
#define VI_PXI_LBUS_STAR_TRIG_BUS_4    (1004)
#define VI_PXI_LBUS_STAR_TRIG_BUS_5    (1005)
#define VI_PXI_LBUS_STAR_TRIG_BUS_6    (1006)
#define VI_PXI_LBUS_STAR_TRIG_BUS_7    (1007)
#define VI_PXI_LBUS_STAR_TRIG_BUS_8    (1008)
#define VI_PXI_LBUS_STAR_TRIG_BUS_9    (1009)
#define VI_PXI_STAR_TRIG_CONTROLLER    (1413)

/*- Backward Compatibility Macros -----*/

#define viGetDefaultRM(vi)             viOpenDefaultRM(vi)
#define VI_ERROR_INV_SESSION           (VI_ERROR_INV_OBJECT)
#define VI_INFINITE                     (VI_TMO_INFINITE)
#define VI_NORMAL                       (VI_PROT_NORMAL)
#define VI_FDC                          (VI_PROT_FDC)
#define VI_HS488                        (VI_PROT_HS488)
#define VI_ASRL488                      (VI_PROT_4882_STRS)
#define VI_ASRL_IN_BUF                  (VI_IO_IN_BUF)
#define VI_ASRL_OUT_BUF                 (VI_IO_OUT_BUF)
#define VI_ASRL_IN_BUF_DISCARD          (VI_IO_IN_BUF_DISCARD)
#define VI_ASRL_OUT_BUF_DISCARD         (VI_IO_OUT_BUF_DISCARD)

#if defined(__cplusplus) || defined(__cplusplus_)
}
#endif
#endif

/*- The End -----*/

```

A.3 Contents of visa32.bas File

This file reflects the required implementation of the specifications given in this document.

```

' -----
'   Distributed by VXIplug&play Systems Alliance
'   Do not modify the contents of this file.
' -----
'   Title   : VISA32.BAS
'   Date    : 06-08-2010
'   Purpose : Include file for the VISA Library 5.0 spec
' -----

Global Const VI_SPEC_VERSION = &H00400000&

' - Resource Template Functions and Operations -----

Declare Function viOpenDefaultRM Lib "VISA32.DLL" Alias "#141" (sesn As Long) As Long
Declare Function viGetDefaultRM Lib "VISA32.DLL" Alias "#128" (sesn As Long) As Long
Declare Function viFindRsrc Lib "VISA32.DLL" Alias "#129" (ByVal sesn As Long, ByVal expr As
String, vi As Long, retCount As Long, ByVal desc As String) As Long
Declare Function viFindNext Lib "VISA32.DLL" Alias "#130" (ByVal vi As Long, ByVal desc As
String) As Long
Declare Function viParseRsrc Lib "VISA32.DLL" Alias "#146" (ByVal sesn As Long, ByVal desc As
String, intfType As Integer, intfNum As Integer) As Long
Declare Function viParseRsrcEx Lib "VISA32.DLL" Alias "#147" (ByVal sesn As Long, ByVal desc As
String, intfType As Integer, intfNum As Integer, ByVal rsrcClass As String,
ByVal expandedUnaliasedName As String, ByVal aliasIfExists As String) As Long
Declare Function viOpen Lib "VISA32.DLL" Alias "#131" (ByVal sesn As Long, ByVal viDesc As
String, ByVal mode As Long, ByVal timeout As Long, vi As Long) As Long
Declare Function viClose Lib "VISA32.DLL" Alias "#132" (ByVal vi As Long) As Long
Declare Function viGetAttribute Lib "VISA32.DLL" Alias "#133" (ByVal vi As Long, ByVal attrName
As Long, attrValue As Any) As Long
Declare Function viSetAttribute Lib "VISA32.DLL" Alias "#134" (ByVal vi As Long, ByVal attrName
As Long, ByVal attrValue As Long) As Long
Declare Function viStatusDesc Lib "VISA32.DLL" Alias "#142" (ByVal vi As Long, ByVal status As
Long, ByVal desc As String) As Long
Declare Function viLock Lib "VISA32.DLL" Alias "#144" (ByVal vi As Long, ByVal lockType As Long,
ByVal timeout As Long, ByVal requestedKey As String, ByVal accessKey As
String) As Long
Declare Function viUnlock Lib "VISA32.DLL" Alias "#145" (ByVal vi As Long) As Long
Declare Function viEnableEvent Lib "VISA32.DLL" Alias "#135" (ByVal vi As Long, ByVal eventType
As Long, ByVal mechanism As Integer, ByVal context As Long) As Long
Declare Function viDisableEvent Lib "VISA32.DLL" Alias "#136" (ByVal vi As Long, ByVal eventType
As Long, ByVal mechanism As Integer) As Long
Declare Function viDiscardEvents Lib "VISA32.DLL" Alias "#137" (ByVal vi As Long, ByVal eventType
As Long, ByVal mechanism As Integer) As Long
Declare Function viWaitOnEvent Lib "VISA32.DLL" Alias "#138" (ByVal vi As Long, ByVal inEventType
As Long, ByVal timeout As Long, outEventType As Long, outEventContext As
Long) As Long

' - Basic I/O Operations -----

Declare Function viRead Lib "VISA32.DLL" Alias "#256" (ByVal vi As Long, ByVal Buffer As String,
ByVal count As Long, retCount As Long) As Long
Declare Function viReadToFile Lib "VISA32.DLL" Alias "#219" (ByVal vi As Long, ByVal filename As
String, ByVal count As Long, retCount As Long) As Long
Declare Function viWrite Lib "VISA32.DLL" Alias "#257" (ByVal vi As Long, ByVal Buffer As String,
ByVal count As Long, retCount As Long) As Long
Declare Function viWriteFromFile Lib "VISA32.DLL" Alias "#218" (ByVal vi As Long, ByVal filename
As String, ByVal count As Long, retCount As Long) As Long
Declare Function viAssertTrigger Lib "VISA32.DLL" Alias "#258" (ByVal vi As Long, ByVal protocol
As Integer) As Long
Declare Function viReadSTB Lib "VISA32.DLL" Alias "#259" (ByVal vi As Long, status As Integer) As
Long
Declare Function viClear Lib "VISA32.DLL" Alias "#260" (ByVal vi As Long) As Long

' - Formatted and Buffered I/O Operations -----

Declare Function viSetBuf Lib "VISA32.DLL" Alias "#267" (ByVal vi As Long, ByVal mask As Integer,
ByVal bufSize As Long) As Long
Declare Function viFlush Lib "VISA32.DLL" Alias "#268" (ByVal vi As Long, ByVal mask As Integer)
As Long

```

```

Declare Function viBufWrite Lib "VISA32.DLL" Alias "#202" (ByVal vi As Long, ByVal Buffer As
String, ByVal count As Long, retCount As Long) As Long
Declare Function viBufRead Lib "VISA32.DLL" Alias "#203" (ByVal vi As Long, ByVal Buffer As
String, ByVal count As Long, retCount As Long) As Long
Declare Function viVPrintf Lib "VISA32.DLL" Alias "#270" (ByVal vi As Long, ByVal writeFmt As
String, params As Any) As Long
Declare Function viVSPrintf Lib "VISA32.DLL" Alias "#205" (ByVal vi As Long, ByVal Buffer As
String, ByVal writeFmt As String, params As Any) As Long
Declare Function viVScanf Lib "VISA32.DLL" Alias "#272" (ByVal vi As Long, ByVal readFmt As
String, params As Any) As Long
Declare Function viVSScanf Lib "VISA32.DLL" Alias "#207" (ByVal vi As Long, ByVal Buffer As
String, ByVal readFmt As String, params As Any) As Long
Declare Function viVQueryf Lib "VISA32.DLL" Alias "#280" (ByVal vi As Long, ByVal writeFmt As
String, ByVal readFmt As String, params As Any) As Long

' - Memory I/O Operations -----
Declare Function viIn8 Lib "VISA32.DLL" Alias "#273" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, val8 As Byte) As Long
Declare Function viOut8 Lib "VISA32.DLL" Alias "#274" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal val8 As Byte) As Long
Declare Function viIn16 Lib "VISA32.DLL" Alias "#261" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, val16 As Integer) As Long
Declare Function viOut16 Lib "VISA32.DLL" Alias "#262" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal val16 As Integer) As Long
Declare Function viIn32 Lib "VISA32.DLL" Alias "#281" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, val32 As Long) As Long
Declare Function viOut32 Lib "VISA32.DLL" Alias "#282" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal val32 As Long) As Long
Declare Function viMoveIn8 Lib "VISA32.DLL" Alias "#283" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal length As Long, buf8 As Byte) As Long
Declare Function viMoveOut8 Lib "VISA32.DLL" Alias "#284" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal length As Long, buf8 As Byte) As Long
Declare Function viMoveIn16 Lib "VISA32.DLL" Alias "#285" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal length As Long, buf16 As Integer) As
Long
Declare Function viMoveOut16 Lib "VISA32.DLL" Alias "#286" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal length As Long, buf16 As Integer) As
Long
Declare Function viMoveIn32 Lib "VISA32.DLL" Alias "#287" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal length As Long, buf32 As Long) As Long
Declare Function viMoveOut32 Lib "VISA32.DLL" Alias "#288" (ByVal vi As Long, ByVal accSpace As
Integer, ByVal offset As Long, ByVal length As Long, buf32 As Long) As Long
Declare Function viMove Lib "VISA32.DLL" Alias "#200" (ByVal vi As Long, ByVal srcSpace As
Integer, ByVal srcOffset As Long, ByVal srcWidth As Integer, ByVal destSpace
As Integer, ByVal destOffset As Long, ByVal destWidth As Integer, ByVal
srcLength As Long) As Long
Declare Function viMapAddress Lib "VISA32.DLL" Alias "#263" (ByVal vi As Long, ByVal mapSpace As
Integer, ByVal mapOffset As Long, ByVal mapSize As Long, ByVal accMode As
Integer, ByVal suggested As Long, address As Long) As Long
Declare Function viUnmapAddress Lib "VISA32.DLL" Alias "#264" (ByVal vi As Long) As Long
Declare Sub viPeek8 Lib "VISA32.DLL" Alias "#275" (ByVal vi As Long, ByVal address As Long, val8
As Byte)
Declare Sub viPoke8 Lib "VISA32.DLL" Alias "#276" (ByVal vi As Long, ByVal address As Long, ByVal
val8 As Byte)
Declare Sub viPeek16 Lib "VISA32.DLL" Alias "#265" (ByVal vi As Long, ByVal address As Long,
val16 As Integer)
Declare Sub viPoke16 Lib "VISA32.DLL" Alias "#266" (ByVal vi As Long, ByVal address As Long,
ByVal val16 As Integer)
Declare Sub viPeek32 Lib "VISA32.DLL" Alias "#289" (ByVal vi As Long, ByVal address As Long,
val32 As Long)
Declare Sub viPoke32 Lib "VISA32.DLL" Alias "#290" (ByVal vi As Long, ByVal address As Long,
ByVal val32 As Long)

' - Shared Memory Operations -----
Declare Function viMemAlloc Lib "VISA32.DLL" Alias "#291" (ByVal vi As Long, ByVal memSize As
Long, offset As Long) As Long
Declare Function viMemFree Lib "VISA32.DLL" Alias "#292" (ByVal vi As Long, ByVal offset As Long)
As Long

' - Interface Specific Operations -----
Declare Function viGpibControlREN Lib "VISA32.DLL" Alias "#208" (ByVal vi As Long, ByVal mode As
Integer) As Long

```

```

Declare Function viGpibControlATN Lib "VISA32.DLL" Alias "#210" (ByVal vi As Long, ByVal mode As Integer) As Long
Declare Function viGpibSendIFC Lib "VISA32.DLL" Alias "#211" (ByVal vi As Long) As Long
Declare Function viGpibCommand Lib "VISA32.DLL" Alias "#212" (ByVal vi As Long, ByVal Buffer As String, ByVal count As Long, retCount As Long) As Long
Declare Function viGpibPassControl Lib "VISA32.DLL" Alias "#213" (ByVal vi As Long, ByVal primAddr As Integer, ByVal secAddr As Integer) As Long
Declare Function viVxiCommandQuery Lib "VISA32.DLL" Alias "#209" (ByVal vi As Long, ByVal mode As Integer, ByVal devCmd As Long, devResponse As Long) As Long
Declare Function viAssertUtilSignal Lib "VISA32.DLL" Alias "#214" (ByVal vi As Long, ByVal line As Integer) As Long
Declare Function viAssertIntrSignal Lib "VISA32.DLL" Alias "#215" (ByVal vi As Long, ByVal mode As Integer, ByVal statusID As Long) As Long
Declare Function viMapTrigger Lib "VISA32.DLL" Alias "#216" (ByVal vi As Long, ByVal trigSrc As Integer, ByVal trigDest As Integer, ByVal mode As Integer) As Long
Declare Function viUnmapTrigger Lib "VISA32.DLL" Alias "#217" (ByVal vi As Long, ByVal trigSrc As Integer, ByVal trigDest As Integer) As Long
Declare Function viUsbControlOut Lib "VISA32.DLL" Alias "#293" (ByVal vi As Long, ByVal bmRequestType As Integer, ByVal bRequest As Integer, ByVal wValue As Integer, ByVal wIndex As Integer, ByVal wLength As Integer, buf As Byte) As Long
Declare Function viUsbControlIn Lib "VISA32.DLL" Alias "#294" (ByVal vi As Long, ByVal bmRequestType As Integer, ByVal bRequest As Integer, ByVal wValue As Integer, ByVal wIndex As Integer, ByVal wLength As Integer, buf As Byte, retCnt As Integer) As Long

```

' - Attributes -----

```

Global Const VI_ATTR_RSRC_CLASS = &HBFFF0001&
Global Const VI_ATTR_RSRC_NAME = &HBFFF0002&
Global Const VI_ATTR_RSRC_IMPL_VERSION = &H3FFF0003&
Global Const VI_ATTR_RSRC_LOCK_STATE = &H3FFF0004&
Global Const VI_ATTR_MAX_QUEUE_LENGTH = &H3FFF0005&
Global Const VI_ATTR_USER_DATA = &H3FFF0007&
Global Const VI_ATTR_USER_DATA_32 = &H3FFF0007&
Global Const VI_ATTR_FDC_CHNL = &H3FFF000D&
Global Const VI_ATTR_FDC_MODE = &H3FFF000F&
Global Const VI_ATTR_FDC_GEN_SIGNAL_EN = &H3FFF0011&
Global Const VI_ATTR_FDC_USE_PAIR = &H3FFF0013&
Global Const VI_ATTR_SEND_END_EN = &H3FFF0016&
Global Const VI_ATTR_TERMCHAR = &H3FFF0018&
Global Const VI_ATTR_TMO_VALUE = &H3FFF001A&
Global Const VI_ATTR_GPIB_READDR_EN = &H3FFF001B&
Global Const VI_ATTR_IO_PROT = &H3FFF001C&
Global Const VI_ATTR_DMA_ALLOW_EN = &H3FFF001E&
Global Const VI_ATTR_ASRL_BAUD = &H3FFF0021&
Global Const VI_ATTR_ASRL_DATA_BITS = &H3FFF0022&
Global Const VI_ATTR_ASRL_PARITY = &H3FFF0023&
Global Const VI_ATTR_ASRL_STOP_BITS = &H3FFF0024&
Global Const VI_ATTR_ASRL_FLOW_CNTRL = &H3FFF0025&
Global Const VI_ATTR_RD_BUF_OPER_MODE = &H3FFF002A&
Global Const VI_ATTR_RD_BUF_SIZE = &H3FFF002B&
Global Const VI_ATTR_WR_BUF_OPER_MODE = &H3FFF002D&
Global Const VI_ATTR_WR_BUF_SIZE = &H3FFF002E&
Global Const VI_ATTR_SUPPRESS_END_EN = &H3FFF0036&
Global Const VI_ATTR_TERMCHAR_EN = &H3FFF0038&
Global Const VI_ATTR_DEST_ACCESS_PRIV = &H3FFF0039&
Global Const VI_ATTR_DEST_BYTE_ORDER = &H3FFF003A&
Global Const VI_ATTR_SRC_ACCESS_PRIV = &H3FFF003C&
Global Const VI_ATTR_SRC_BYTE_ORDER = &H3FFF003D&
Global Const VI_ATTR_SRC_INCREMENT = &H3FFF0040&
Global Const VI_ATTR_DEST_INCREMENT = &H3FFF0041&
Global Const VI_ATTR_WIN_ACCESS_PRIV = &H3FFF0045&
Global Const VI_ATTR_WIN_BYTE_ORDER = &H3FFF0047&
Global Const VI_ATTR_GPIB_ATN_STATE = &H3FFF0057&
Global Const VI_ATTR_GPIB_ADDR_STATE = &H3FFF005C&
Global Const VI_ATTR_GPIB_CIC_STATE = &H3FFF005E&
Global Const VI_ATTR_GPIB_NDAC_STATE = &H3FFF0062&
Global Const VI_ATTR_GPIB_SRQ_STATE = &H3FFF0067&
Global Const VI_ATTR_GPIB_SYS_CNTRL_STATE = &H3FFF0068&
Global Const VI_ATTR_GPIB_HS488_CBL_LEN = &H3FFF0069&
Global Const VI_ATTR_CMDR_LA = &H3FFF006B&
Global Const VI_ATTR_VXI_DEV_CLASS = &H3FFF006C&
Global Const VI_ATTR_MAINFRAME_LA = &H3FFF0070&
Global Const VI_ATTR_MANF_NAME = &HBFFF0072&
Global Const VI_ATTR_MODEL_NAME = &HBFFF0077&
Global Const VI_ATTR_VXI_VME_INTR_STATUS = &H3FFF008B&
Global Const VI_ATTR_VXI_TRIG_STATUS = &H3FFF008D&

```

```

Global Const VI_ATTR_VXI_VME_SYSFAIL_STATE = &H3FFF0094&
Global Const VI_ATTR_WIN_BASE_ADDR = &H3FFF0098&
Global Const VI_ATTR_WIN_BASE_ADDR_32 = &H3FFF0098&
Global Const VI_ATTR_WIN_SIZE = &H3FFF009A&
Global Const VI_ATTR_WIN_SIZE_32 = &H3FFF009A&
Global Const VI_ATTR_ASRL_AVAIL_NUM = &H3FFF00AC&
Global Const VI_ATTR_MEM_BASE = &H3FFF00AD&
Global Const VI_ATTR_MEM_BASE_32 = &H3FFF00AD&
Global Const VI_ATTR_ASRL_CTS_STATE = &H3FFF00AE&
Global Const VI_ATTR_ASRL_DCD_STATE = &H3FFF00AF&
Global Const VI_ATTR_ASRL_DSR_STATE = &H3FFF00B1&
Global Const VI_ATTR_ASRL_DTR_STATE = &H3FFF00B2&
Global Const VI_ATTR_ASRL_END_IN = &H3FFF00B3&
Global Const VI_ATTR_ASRL_END_OUT = &H3FFF00B4&
Global Const VI_ATTR_ASRL_REPLACE_CHAR = &H3FFF00BE&
Global Const VI_ATTR_ASRL_RI_STATE = &H3FFF00BF&
Global Const VI_ATTR_ASRL_RTS_STATE = &H3FFF00C0&
Global Const VI_ATTR_ASRL_XON_CHAR = &H3FFF00C1&
Global Const VI_ATTR_ASRL_XOFF_CHAR = &H3FFF00C2&
Global Const VI_ATTR_WIN_ACCESS = &H3FFF00C3&
Global Const VI_ATTR_RM_SESSION = &H3FFF00C4&
Global Const VI_ATTR_VXI_LA = &H3FFF00D5&
Global Const VI_ATTR_MANF_ID = &H3FFF00D9&
Global Const VI_ATTR_MEM_SIZE = &H3FFF00DD&
Global Const VI_ATTR_MEM_SIZE_32 = &H3FFF00DD&
Global Const VI_ATTR_MEM_SPACE = &H3FFF00DE&
Global Const VI_ATTR_MODEL_CODE = &H3FFF00DF&
Global Const VI_ATTR_SLOT = &H3FFF00E8&
Global Const VI_ATTR_INTF_INST_NAME = &HBFFF00E9&
Global Const VI_ATTR_IMMEDIATE_SERV = &H3FFF0100&
Global Const VI_ATTR_INTF_PARENT_NUM = &H3FFF0101&
Global Const VI_ATTR_RSRC_SPEC_VERSION = &H3FFF0170&
Global Const VI_ATTR_INTF_TYPE = &H3FFF0171&
Global Const VI_ATTR_GPIB_PRIMARY_ADDR = &H3FFF0172&
Global Const VI_ATTR_GPIB_SECONDARY_ADDR = &H3FFF0173&
Global Const VI_ATTR_RSRC_MANF_NAME = &HBFFF0174&
Global Const VI_ATTR_RSRC_MANF_ID = &H3FFF0175&
Global Const VI_ATTR_INTF_NUM = &H3FFF0176&
Global Const VI_ATTR_TRIG_ID = &H3FFF0177&
Global Const VI_ATTR_GPIB_REN_STATE = &H3FFF0181&
Global Const VI_ATTR_GPIB_UNADDR_EN = &H3FFF0184&
Global Const VI_ATTR_DEV_STATUS_BYTE = &H3FFF0189&
Global Const VI_ATTR_FILE_APPEND_EN = &H3FFF0192&
Global Const VI_ATTR_VXI_TRIG_SUPPORT = &H3FFF0194&
Global Const VI_ATTR_TCPIP_ADDR = &HBFFF0195&
Global Const VI_ATTR_TCPIP_HOSTNAME = &HBFFF0196&
Global Const VI_ATTR_TCPIP_PORT = &H3FFF0197&
Global Const VI_ATTR_TCPIP_DEVICE_NAME = &HBFFF0199&
Global Const VI_ATTR_TCPIP_NODELAY = &H3FFF019A&
Global Const VI_ATTR_TCPIP_KEEPALIVE = &H3FFF019B&
Global Const VI_ATTR_4882_COMPLIANT = &H3FFF019F&
Global Const VI_ATTR_USB_SERIAL_NUM = &HBFFF01A0&
Global Const VI_ATTR_USB_INTFC_NUM = &H3FFF01A1&
Global Const VI_ATTR_USB_PROTOCOL = &H3FFF01A7&
Global Const VI_ATTR_USB_MAX_INTR_SIZE = &H3FFF01AF&

Global Const VI_ATTR_PXI_DEV_NUM = &H3FFF0201&
Global Const VI_ATTR_PXI_FUNC_NUM = &H3FFF0202&
Global Const VI_ATTR_PXI_BUS_NUM = &H3FFF0205&
Global Const VI_ATTR_PXI_CHASSIS = &H3FFF0206&
Global Const VI_ATTR_PXI_SLOTPATH = &HBFFF0207&
Global Const VI_ATTR_PXI_SLOT_LBUS_LEFT = &H3FFF0208&
Global Const VI_ATTR_PXI_SLOT_LBUS_RIGHT = &H3FFF0209&
Global Const VI_ATTR_PXI_TRIG_BUS = &H3FFF020A&
Global Const VI_ATTR_PXI_STAR_TRIG_BUS = &H3FFF020B&
Global Const VI_ATTR_PXI_STAR_TRIG_LINE = &H3FFF020C&
Global Const VI_ATTR_PXI_MEM_TYPE_BAR0 = &H3FFF0211&
Global Const VI_ATTR_PXI_MEM_TYPE_BAR1 = &H3FFF0212&
Global Const VI_ATTR_PXI_MEM_TYPE_BAR2 = &H3FFF0213&
Global Const VI_ATTR_PXI_MEM_TYPE_BAR3 = &H3FFF0214&
Global Const VI_ATTR_PXI_MEM_TYPE_BAR4 = &H3FFF0215&
Global Const VI_ATTR_PXI_MEM_TYPE_BAR5 = &H3FFF0216&
Global Const VI_ATTR_PXI_MEM_BASE_BAR0 = &H3FFF0221&
Global Const VI_ATTR_PXI_MEM_BASE_BAR1 = &H3FFF0222&
Global Const VI_ATTR_PXI_MEM_BASE_BAR2 = &H3FFF0223&
Global Const VI_ATTR_PXI_MEM_BASE_BAR3 = &H3FFF0224&
Global Const VI_ATTR_PXI_MEM_BASE_BAR4 = &H3FFF0225&

```

```

Global Const VI_ATTR_PXI_MEM_BASE_BAR5 = &H3FFF0226&
Global Const VI_ATTR_PXI_MEM_SIZE_BAR0 = &H3FFF0231&
Global Const VI_ATTR_PXI_MEM_SIZE_BAR1 = &H3FFF0232&
Global Const VI_ATTR_PXI_MEM_SIZE_BAR2 = &H3FFF0233&
Global Const VI_ATTR_PXI_MEM_SIZE_BAR3 = &H3FFF0234&
Global Const VI_ATTR_PXI_MEM_SIZE_BAR4 = &H3FFF0235&
Global Const VI_ATTR_PXI_MEM_SIZE_BAR5 = &H3FFF0236&
Global Const VI_ATTR_PXI_IS_EXPRESS = &H3FFF0240&
Global Const VI_ATTR_PXI_SLOT_LWIDTH = &H3FFF0241&
Global Const VI_ATTR_PXI_MAX_LWIDTH = &H3FFF0242&
Global Const VI_ATTR_PXI_ACTUAL_LWIDTH = &H3FFF0243&
Global Const VI_ATTR_PXI_DSTAR_BUS = &H3FFF0244&
Global Const VI_ATTR_PXI_DSTAR_SET = &H3FFF0245&

Global Const VI_ATTR_TCPIP_HISLIP_OVERLAP_EN = &H3FFF0300&
Global Const VI_ATTR_TCPIP_HISLIP_VERSION = &H3FFF0301&
Global Const VI_ATTR_TCPIP_HISLIP_MAX_MESSAGE_KB = &H3FFF0302&

```

```

Global Const VI_ATTR_JOB_ID = &H3FFF4006&
Global Const VI_ATTR_EVENT_TYPE = &H3FFF4010&
Global Const VI_ATTR_SIGP_STATUS_ID = &H3FFF4011&
Global Const VI_ATTR_RECV_TRIG_ID = &H3FFF4012&
Global Const VI_ATTR_INTR_STATUS_ID = &H3FFF4023&
Global Const VI_ATTR_STATUS = &H3FFF4025&
Global Const VI_ATTR_RET_COUNT = &H3FFF4026&
Global Const VI_ATTR_RET_COUNT_32 = &H3FFF4026&
Global Const VI_ATTR_BUFFER = &H3FFF4027&
Global Const VI_ATTR_RECV_INTR_LEVEL = &H3FFF4041&
Global Const VI_ATTR_OPER_NAME = &HBFFF4042&
Global Const VI_ATTR_GPIB_RECV_CIC_STATE = &H3FFF4193&
Global Const VI_ATTR_RECV_TCPIP_ADDR = &HBFFF4198&
Global Const VI_ATTR_USB_RECV_INTR_SIZE = &H3FFF41B0&
Global Const VI_ATTR_USB_RECV_INTR_DATA = &HBFFF41B1&

```

' - Event Types -----

```

Global Const VI_EVENT_IO_COMPLETION = &H3FFF2009&
Global Const VI_EVENT_TRIG = &HBFFF200A&
Global Const VI_EVENT_SERVICE_REQ = &H3FFF200B&
Global Const VI_EVENT_CLEAR = &H3FFF200D&
Global Const VI_EVENT_EXCEPTION = &HBFFF200E&
Global Const VI_EVENT_GPIB_CIC = &H3FFF2012&
Global Const VI_EVENT_GPIB_TALK = &H3FFF2013&
Global Const VI_EVENT_GPIB_LISTEN = &H3FFF2014&
Global Const VI_EVENT_VXI_VME_SYSFAIL = &H3FFF201D&
Global Const VI_EVENT_VXI_VME_SYSRESET = &H3FFF201E&
Global Const VI_EVENT_VXI_SIGP = &H3FFF2020&
Global Const VI_EVENT_VXI_VME_INTR = &HBFFF2021&
Global Const VI_EVENT_TCPIP_CONNECT = &H3FFF2036&
Global Const VI_EVENT_USB_INTR = &H3FFF2037&

Global Const VI_EVENT_PXI_INTR = &H3FFF2022&

Global Const VI_ALL_ENABLED_EVENTS = &H3FFF7FFF&

```

' - Completion and Error Codes -----

```

Global Const VI_SUCCESS = &H0&
Global Const VI_SUCCESS_EVENT_EN = &H3FFF0002&
Global Const VI_SUCCESS_EVENT_DIS = &H3FFF0003&
Global Const VI_SUCCESS_QUEUE_EMPTY = &H3FFF0004&
Global Const VI_SUCCESS_TERM_CHAR = &H3FFF0005&
Global Const VI_SUCCESS_MAX_CNT = &H3FFF0006&
Global Const VI_SUCCESS_DEV_NPRESENT = &H3FFF007D&
Global Const VI_SUCCESS_TRIG_MAPPED = &H3FFF007E&
Global Const VI_SUCCESS_QUEUE_NEMPTY = &H3FFF0080&
Global Const VI_SUCCESS_NCHAIN = &H3FFF0098&
Global Const VI_SUCCESS_NESTED_SHARED = &H3FFF0099&
Global Const VI_SUCCESS_NESTED_EXCLUSIVE = &H3FFF009A&
Global Const VI_SUCCESS_SYNC = &H3FFF009B&

Global Const VI_WARN_QUEUE_OVERFLOW = &H3FFF000C&
Global Const VI_WARN_CONFIG_NLOADED = &H3FFF0077&
Global Const VI_WARN_NULL_OBJECT = &H3FFF0082&
Global Const VI_WARN_NSUP_ATTR_STATE = &H3FFF0084&
Global Const VI_WARN_UNKNOWN_STATUS = &H3FFF0085&

```

```

Global Const VI_WARN_NSUP_BUF = &H3FFF0088&
Global Const VI_WARN_EXT_FUNC_NIMPL = &H3FFF00A9&

Global Const VI_ERROR_SYSTEM_ERROR = &HBFFF0000&
Global Const VI_ERROR_INV_OBJECT = &HBFFF000E&
Global Const VI_ERROR_RSRC_LOCKED = &HBFFF000F&
Global Const VI_ERROR_INV_EXPR = &HBFFF0010&
Global Const VI_ERROR_RSRC_NFOUND = &HBFFF0011&
Global Const VI_ERROR_INV_RSRC_NAME = &HBFFF0012&
Global Const VI_ERROR_INV_ACC_MODE = &HBFFF0013&
Global Const VI_ERROR_TMO = &HBFFF0015&
Global Const VI_ERROR_CLOSING_FAILED = &HBFFF0016&
Global Const VI_ERROR_INV_DEGREE = &HBFFF001B&
Global Const VI_ERROR_INV_JOB_ID = &HBFFF001C&
Global Const VI_ERROR_NSUP_ATTR = &HBFFF001D&
Global Const VI_ERROR_NSUP_ATTR_STATE = &HBFFF001E&
Global Const VI_ERROR_ATTR_READONLY = &HBFFF001F&
Global Const VI_ERROR_INV_LOCK_TYPE = &HBFFF0020&
Global Const VI_ERROR_INV_ACCESS_KEY = &HBFFF0021&
Global Const VI_ERROR_INV_EVENT = &HBFFF0026&
Global Const VI_ERROR_INV_MECH = &HBFFF0027&
Global Const VI_ERROR_HNDLR_NINSTALLED = &HBFFF0028&
Global Const VI_ERROR_INV_HNDLR_REF = &HBFFF0029&
Global Const VI_ERROR_INV_CONTEXT = &HBFFF002A&
Global Const VI_ERROR_NENABLED = &HBFFF002F&
Global Const VI_ERROR_ABORT = &HBFFF0030&
Global Const VI_ERROR_RAW_WR_PROT_VIOL = &HBFFF0034&
Global Const VI_ERROR_RAW_RD_PROT_VIOL = &HBFFF0035&
Global Const VI_ERROR_OUTP_PROT_VIOL = &HBFFF0036&
Global Const VI_ERROR_INP_PROT_VIOL = &HBFFF0037&
Global Const VI_ERROR_BERR = &HBFFF0038&
Global Const VI_ERROR_IN_PROGRESS = &HBFFF0039&
Global Const VI_ERROR_INV_SETUP = &HBFFF003A&
Global Const VI_ERROR_QUEUE_ERROR = &HBFFF003B&
Global Const VI_ERROR_ALLOC = &HBFFF003C&
Global Const VI_ERROR_INV_MASK = &HBFFF003D&
Global Const VI_ERROR_IO = &HBFFF003E&
Global Const VI_ERROR_INV_FMT = &HBFFF003F&
Global Const VI_ERROR_NSUP_FMT = &HBFFF0041&
Global Const VI_ERROR_LINE_IN_USE = &HBFFF0042&
Global Const VI_ERROR_NSUP_MODE = &HBFFF0046&
Global Const VI_ERROR_SRQ_NOCCURRED = &HBFFF004A&
Global Const VI_ERROR_INV_SPACE = &HBFFF004E&
Global Const VI_ERROR_INV_OFFSET = &HBFFF0051&
Global Const VI_ERROR_INV_WIDTH = &HBFFF0052&
Global Const VI_ERROR_NSUP_OFFSET = &HBFFF0054&
Global Const VI_ERROR_NSUP_VAR_WIDTH = &HBFFF0055&
Global Const VI_ERROR_WINDOW_NMAPPED = &HBFFF0057&
Global Const VI_ERROR_RESP_PENDING = &HBFFF0059&
Global Const VI_ERROR_NLISTENERS = &HBFFF005F&
Global Const VI_ERROR_NCIC = &HBFFF0060&
Global Const VI_ERROR_NSYS_CNTRLR = &HBFFF0061&
Global Const VI_ERROR_NSUP_OPER = &HBFFF0067&
Global Const VI_ERROR_INTR_PENDING = &HBFFF0068&
Global Const VI_ERROR_ASRL_PARITY = &HBFFF006A&
Global Const VI_ERROR_ASRL_FRAMING = &HBFFF006B&
Global Const VI_ERROR_ASRL_OVERRUN = &HBFFF006C&
Global Const VI_ERROR_TRIG_NMAPPED = &HBFFF006E&
Global Const VI_ERROR_NSUP_ALIGN_OFFSET = &HBFFF0070&
Global Const VI_ERROR_USER_BUF = &HBFFF0071&
Global Const VI_ERROR_RSRC_BUSY = &HBFFF0072&
Global Const VI_ERROR_NSUP_WIDTH = &HBFFF0076&
Global Const VI_ERROR_INV_PARAMETER = &HBFFF0078&
Global Const VI_ERROR_INV_PROT = &HBFFF0079&
Global Const VI_ERROR_INV_SIZE = &HBFFF007B&
Global Const VI_ERROR_WINDOW_MAPPED = &HBFFF0080&
Global Const VI_ERROR_NIMPL_OPER = &HBFFF0081&
Global Const VI_ERROR_INV_LENGTH = &HBFFF0083&
Global Const VI_ERROR_INV_MODE = &HBFFF0091&
Global Const VI_ERROR_SESN_NLOCKED = &HBFFF009C&
Global Const VI_ERROR_MEM_NSHARED = &HBFFF009D&
Global Const VI_ERROR_LIBRARY_NFOUND = &HBFFF009E&
Global Const VI_ERROR_NSUP_INTR = &HBFFF009F&
Global Const VI_ERROR_INV_LINE = &HBFFF00A0&
Global Const VI_ERROR_FILE_ACCESS = &HBFFF00A1&
Global Const VI_ERROR_FILE_IO = &HBFFF00A2&
Global Const VI_ERROR_NSUP_LINE = &HBFFF00A3&

```

```

Global Const VI_ERROR_NSUP_MECH           = &HBFFF00A4&
Global Const VI_ERROR_INTF_NUM_NCONFIG   = &HBFFF00A5&
Global Const VI_ERROR_CONN_LOST         = &HBFFF00A6&
Global Const VI_ERROR_NPERMISSION       = &HBFFF00A8&

```

' - Other VISA Definitions -----

```

Global Const VI_FIND_BUFLEN              = 256

Global Const VI_NULL                     = 0
Global Const VI_TRUE                     = 1
Global Const VI_FALSE                    = 0

Global Const VI_INTF_GPIB                = 1
Global Const VI_INTF_VXI                 = 2
Global Const VI_INTF_GPIB_VXI           = 3
Global Const VI_INTF_ASRL                = 4

Global Const VI_INTF_PXI                  = 5
Global Const VI_INTF_TCPIP               = 6
Global Const VI_INTF_USB                  = 7

Global Const VI_PROT_NORMAL               = 1
Global Const VI_PROT_FDC                 = 2
Global Const VI_PROT_HS488               = 3
Global Const VI_PROT_4882_STRS           = 4
Global Const VI_PROT_USBTMC_VENDOR       = 5

Global Const VI_FDC_NORMAL                = 1
Global Const VI_FDC_STREAM                = 2

Global Const VI_LOCAL_SPACE               = 0
Global Const VI_A16_SPACE                 = 1
Global Const VI_A24_SPACE                 = 2
Global Const VI_A32_SPACE                 = 3
Global Const VI_A64_SPACE                 = 4
Global Const VI_PXI_ALLOC_SPACE           = 9
Global Const VI_PXI_CFG_SPACE             = 10
Global Const VI_PXI_BAR0_SPACE            = 11
Global Const VI_PXI_BAR1_SPACE            = 12
Global Const VI_PXI_BAR2_SPACE            = 13
Global Const VI_PXI_BAR3_SPACE            = 14
Global Const VI_PXI_BAR4_SPACE            = 15
Global Const VI_PXI_BAR5_SPACE            = 16
Global Const VI_OPAQUE_SPACE              = &HFFFF

Global Const VI_UNKNOWN_LA                 = -1
Global Const VI_UNKNOWN_SLOT              = -1
Global Const VI_UNKNOWN_LEVEL             = -1
Global Const VI_UNKNOWN_CHASSIS           = -1

Global Const VI_QUEUE                     = 1
Global Const VI_ALL_MECH                  = &HFFFF

Global Const VI_TRIG_ALL                  = -2
Global Const VI_TRIG_SW                   = -1
Global Const VI_TRIG_TTL0                 = 0
Global Const VI_TRIG_TTL1                 = 1
Global Const VI_TRIG_TTL2                 = 2
Global Const VI_TRIG_TTL3                 = 3
Global Const VI_TRIG_TTL4                 = 4
Global Const VI_TRIG_TTL5                 = 5
Global Const VI_TRIG_TTL6                 = 6
Global Const VI_TRIG_TTL7                 = 7
Global Const VI_TRIG_ECL0                 = 8
Global Const VI_TRIG_ECL1                 = 9
Global Const VI_TRIG_PANEL_IN             = 27
Global Const VI_TRIG_PANEL_OUT            = 28

Global Const VI_TRIG_PROT_DEFAULT         = 0
Global Const VI_TRIG_PROT_ON              = 1
Global Const VI_TRIG_PROT_OFF             = 2
Global Const VI_TRIG_PROT_SYNC            = 5
Global Const VI_TRIG_PROT_RESERVE         = 6
Global Const VI_TRIG_PROT_UNRESERVE       = 7

Global Const VI_READ_BUF                  = 1

```

```

Global Const VI_WRITE_BUF = 2
Global Const VI_READ_BUF_DISCARD = 4
Global Const VI_WRITE_BUF_DISCARD = 8
Global Const VI_IO_IN_BUF = 16
Global Const VI_IO_OUT_BUF = 32
Global Const VI_IO_IN_BUF_DISCARD = 64
Global Const VI_IO_OUT_BUF_DISCARD = 128

Global Const VI_FLUSH_ON_ACCESS = 1
Global Const VI_FLUSH_WHEN_FULL = 2
Global Const VI_FLUSH_DISABLE = 3

Global Const VI_NMAPPED = 1
Global Const VI_USE_OPERS = 2
Global Const VI_DEREF_ADDR = 3

Global Const VI_TMO_IMMEDIATE = &H0&
Global Const VI_TMO_INFINITE = &HFFFFFF&

Global Const VI_NO_LOCK = 0
Global Const VI_EXCLUSIVE_LOCK = 1
Global Const VI_SHARED_LOCK = 2
Global Const VI_LOAD_CONFIG = 4

Global Const VI_NO_SEC_ADDR = &HFFFF

Global Const VI_ASRL_PAR_NONE = 0
Global Const VI_ASRL_PAR_ODD = 1
Global Const VI_ASRL_PAR_EVEN = 2
Global Const VI_ASRL_PAR_MARK = 3
Global Const VI_ASRL_PAR_SPACE = 4

Global Const VI_ASRL_STOP_ONE = 10
Global Const VI_ASRL_STOP_ONE5 = 15
Global Const VI_ASRL_STOP_TWO = 20

Global Const VI_ASRL_FLOW_NONE = 0
Global Const VI_ASRL_FLOW_XON_XOFF = 1
Global Const VI_ASRL_FLOW_RTS_CTS = 2
Global Const VI_ASRL_FLOW_DTR_DSR = 4

Global Const VI_ASRL_END_NONE = 0
Global Const VI_ASRL_END_LAST_BIT = 1
Global Const VI_ASRL_END_TERMCHAR = 2
Global Const VI_ASRL_END_BREAK = 3

Global Const VI_STATE_ASSERTED = 1
Global Const VI_STATE_UNASSERTED = 0
Global Const VI_STATE_UNKNOWN = -1

Global Const VI_BIG_ENDIAN = 0
Global Const VI_LITTLE_ENDIAN = 1

Global Const VI_DATA_PRIV = 0
Global Const VI_DATA_NPRIV = 1
Global Const VI_PROG_PRIV = 2
Global Const VI_PROG_NPRIV = 3
Global Const VI_BLCK_PRIV = 4
Global Const VI_BLCK_NPRIV = 5
Global Const VI_D64_PRIV = 6
Global Const VI_D64_NPRIV = 7

Global Const VI_WIDTH_8 = 1
Global Const VI_WIDTH_16 = 2
Global Const VI_WIDTH_32 = 4
Global Const VI_WIDTH_64 = 8

Global Const VI_GPIB_REN_DEASSERT = 0
Global Const VI_GPIB_REN_ASSERT = 1
Global Const VI_GPIB_REN_DEASSERT_GTL = 2
Global Const VI_GPIB_REN_ASSERT_ADDRESS = 3
Global Const VI_GPIB_REN_ASSERT_LLO = 4
Global Const VI_GPIB_REN_ASSERT_ADDRESS_LLO = 5
Global Const VI_GPIB_REN_ADDRESS_GTL = 6

Global Const VI_GPIB_ATN_DEASSERT = 0
Global Const VI_GPIB_ATN_ASSERT = 1

```

```

Global Const VI_GPIB_ATN_DEASSERT_HANDSHAKE = 2
Global Const VI_GPIB_ATN_ASSERT_IMMEDIATE = 3

Global Const VI_GPIB_HS488_DISABLED = 0
Global Const VI_GPIB_HS488_NIMPL = -1

Global Const VI_GPIB_UNADDRESSED = 0
Global Const VI_GPIB_TALKER = 1
Global Const VI_GPIB_LISTENER = 2

Global Const VI_VXI_CMD16 = &H0200
Global Const VI_VXI_CMD16_RESP16 = &H0202
Global Const VI_VXI_RESP16 = &H0002
Global Const VI_VXI_CMD32 = &H0400
Global Const VI_VXI_CMD32_RESP16 = &H0402
Global Const VI_VXI_CMD32_RESP32 = &H0404
Global Const VI_VXI_RESP32 = &H0004

Global Const VI_ASSERT_SIGNAL = -1
Global Const VI_ASSERT_USE_ASSIGNED = 0
Global Const VI_ASSERT_IRQ1 = 1
Global Const VI_ASSERT_IRQ2 = 2
Global Const VI_ASSERT_IRQ3 = 3
Global Const VI_ASSERT_IRQ4 = 4
Global Const VI_ASSERT_IRQ5 = 5
Global Const VI_ASSERT_IRQ6 = 6
Global Const VI_ASSERT_IRQ7 = 7

Global Const VI_UTIL_ASSERT_SYSRESET = 1
Global Const VI_UTIL_ASSERT_SYSFAIL = 2
Global Const VI_UTIL_DEASSERT_SYSFAIL = 3

Global Const VI_VXI_CLASS_MEMORY = 0
Global Const VI_VXI_CLASS_EXTENDED = 1
Global Const VI_VXI_CLASS_MESSAGE = 2
Global Const VI_VXI_CLASS_REGISTER = 3
Global Const VI_VXI_CLASS_OTHER = 4

Global Const VI_PXI_ADDR_NONE = 0
Global Const VI_PXI_ADDR_MEM = 1
Global Const VI_PXI_ADDR_IO = 2
Global Const VI_PXI_ADDR_CFG = 3

Global Const VI_UNKNOWN_TRIG = -1
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_0 = 1000
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_1 = 1001
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_2 = 1002
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_3 = 1003
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_4 = 1004
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_5 = 1005
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_6 = 1006
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_7 = 1007
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_8 = 1008
Global Const VI_PXI_LBUS_STAR_TRIG_BUS_9 = 1009
Global Const VI_PXI_STAR_TRIG_CONTROLLER = 1413

' - Backward Compatibility Macros -----
Global Const VI_ERROR_INV_SESSION = &HBFFFF000E&
Global Const VI_INFINITE = &HFFFFFFFF&

Global Const VI_NORMAL = 1
Global Const VI_FDC = 2
Global Const VI_HS488 = 3
Global Const VI_ASRL488 = 4

Global Const VI_ASRL_IN_BUF = 16
Global Const VI_ASRL_OUT_BUF = 32
Global Const VI_ASRL_IN_BUF_DISCARD = 64
Global Const VI_ASRL_OUT_BUF_DISCARD = 128

```

A.4 Contents of visa32.def File

This file reflects a preferred implementation of the specifications given in this document.

LIBRARY	VISA32	
EXPORTS		
	viGetDefaultRM	@128
	viOpenDefaultRM	@141
	viFindRsrc	@129
	viFindNext	@130
	viOpen	@131
	viClose	@132
	viGetAttribute	@133
	viSetAttribute	@134
	viStatusDesc	@142
	viTerminate	@143
	viLock	@144
	viUnlock	@145
	viEnableEvent	@135
	viDisableEvent	@136
	viDiscardEvents	@137
	viWaitOnEvent	@138
	viInstallHandler	@139
	viUninstallHandler	@140
	viParseRsrc	@146
	viParseRsrcEx	@147
	viMove	@200
	viMoveAsync	@201
	viBufWrite	@202
	viBufRead	@203
	viSPrintf	@204
	viVSPrintf	@205
	viSScanf	@206
	viVSScanf	@207
	viGpibControlREN	@208
	viVxiCommandQuery	@209
	viGpibControlATN	@210
	viGpibSendIFC	@211
	viGpibCommand	@212
	viGpibPassControl	@213
	viAssertUtilSignal	@214
	viAssertIntrSignal	@215
	viMapTrigger	@216
	viUnmapTrigger	@217
	viWriteFromFile	@218
	viReadToFile	@219
	viIn64	@220
	viOut64	@221
	viIn8Ex	@222
	viOut8Ex	@223
	viIn16Ex	@224
	viOut16Ex	@225
	viIn32Ex	@226
	viOut32Ex	@227
	viIn64Ex	@228
	viOut64Ex	@229
	viMoveIn64	@230
	viMoveOut64	@231
	viMoveIn8Ex	@232
	viMoveOut8Ex	@233
	viMoveIn16Ex	@234
	viMoveOut16Ex	@235
	viMoveIn32Ex	@236
	viMoveOut32Ex	@237
	viMoveIn64Ex	@238
	viMoveOut64Ex	@239
	viMoveEx	@240
	viMoveAsyncEx	@241
	viMapAddressEx	@242
	viMemAllocEx	@243
	viMemFreeEx	@244
	viPeek64	@245

viPoke64	@246
viRead	@256
viReadAsync	@277
viWrite	@257
viWriteAsync	@278
viAssertTrigger	@258
viReadSTB	@259
viClear	@260
viSetBuf	@267
viFlush	@268
viPrintf	@269
viVPrintf	@270
viScanf	@271
viVScanf	@272
viQueryf	@279
viVQueryf	@280
viIn8	@273
viOut8	@274
viIn16	@261
viOut16	@262
viIn32	@281
viOut32	@282
viMoveIn8	@283
viMoveOut8	@284
viMoveIn16	@285
viMoveOut16	@286
viMoveIn32	@287
viMoveOut32	@288
viMapAddress	@263
viUnmapAddress	@264
viPeek8	@275
viPoke8	@276
viPeek16	@265
viPoke16	@266
viPeek32	@289
viPoke32	@290
viMemAlloc	@291
viMemFree	@292
viUsbControlOut	@293
viUsbControlIn	@294

A.5 Contents of visa64.def File

This file reflects a preferred implementation of the specifications given in this document.

LIBRARY	VISA64	
EXPORTS		
	viGetDefaultRM	@128
	viOpenDefaultRM	@141
	viFindRsrc	@129
	viFindNext	@130
	viOpen	@131
	viClose	@132
	viGetAttribute	@133
	viSetAttribute	@134
	viStatusDesc	@142
	viTerminate	@143
	viLock	@144
	viUnlock	@145
	viEnableEvent	@135
	viDisableEvent	@136
	viDiscardEvents	@137
	viWaitOnEvent	@138
	viInstallHandler	@139
	viUninstallHandler	@140
	viParseRsrc	@146
	viParseRsrcEx	@147
	viMove	@200
	viMoveAsync	@201
	viBufWrite	@202
	viBufRead	@203
	viSPrintf	@204
	viVSPrintf	@205
	viSScanf	@206
	viVSScanf	@207
	viGpibControlREN	@208
	viVxiCommandQuery	@209
	viGpibControlATN	@210
	viGpibSendIFC	@211
	viGpibCommand	@212
	viGpibPassControl	@213
	viAssertUtilSignal	@214
	viAssertIntrSignal	@215
	viMapTrigger	@216
	viUnmapTrigger	@217
	viWriteFromFile	@218
	viReadToFile	@219
	viIn64	@220
	viOut64	@221
	viIn8Ex	@222
	viOut8Ex	@223
	viIn16Ex	@224
	viOut16Ex	@225
	viIn32Ex	@226
	viOut32Ex	@227
	viIn64Ex	@228
	viOut64Ex	@229
	viMoveIn64	@230
	viMoveOut64	@231
	viMoveIn8Ex	@232
	viMoveOut8Ex	@233
	viMoveIn16Ex	@234
	viMoveOut16Ex	@235
	viMoveIn32Ex	@236
	viMoveOut32Ex	@237
	viMoveIn64Ex	@238
	viMoveOut64Ex	@239
	viMoveEx	@240
	viMoveAsyncEx	@241
	viMapAddressEx	@242
	viMemAllocEx	@243
	viMemFreeEx	@244
	viPeek64	@245

viPoke64	@246
viRead	@256
viReadAsync	@277
viWrite	@257
viWriteAsync	@278
viAssertTrigger	@258
viReadSTB	@259
viClear	@260
viSetBuf	@267
viFlush	@268
viPrintf	@269
viVPrintf	@270
viScanf	@271
viVScanf	@272
viQueryf	@279
viVQueryf	@280
viIn8	@273
viOut8	@274
viIn16	@261
viOut16	@262
viIn32	@281
viOut32	@282
viMoveIn8	@283
viMoveOut8	@284
viMoveIn16	@285
viMoveOut16	@286
viMoveIn32	@287
viMoveOut32	@288
viMapAddress	@263
viUnmapAddress	@264
viPeek8	@275
viPoke8	@276
viPeek16	@265
viPoke16	@266
viPeek32	@289
viPoke32	@290
viMemAlloc	@291
viMemFree	@292
viUsbControlOut	@293
viUsbControlIn	@294