

**Systems Alliance**

**VPP-4.3.2: VISA Implementation Specification for Textual Languages**

**February 26, 2016**

**Revision 5.7**



**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.

**Revision 5.1, October 11, 2012**

Added support extended VXIbus block transfer protocols and trigger capabilities according to VXI-1 4.0. Extensions for PXI.

**Revision 5.4, June 19, 2014**

Added a new error code VI\_ERROR\_LINE\_NRESERVED to facilitate better mapping of PXI-9 trigger error codes. Added support for LCC compiler. Changed the version to 5.4 to ensure that all VISA specs being voted on at the same time have the same version.

**Revision 5.7, February 26, 2016**

Add PXI trigger lines TTL8-TTL11. Added support for MinGW and Clang compilers.

**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*plug&play* Systems Alliance and the IVI Foundation 1-1](#_Toc135113804)

[Section 2 Overview of VISA Implementation Specification 2-1](#_Toc135113805)

[2.1 Objectives of This Specification 2-1](#_Toc135113806)

[2.2 Audience for This Specification 2-1](#_Toc135113807)

[2.3 Scope and Organization of This Specification 2-2](#_Toc135113808)

[2.4 Application of This Specification 2-2](#_Toc135113809)

[2.5 References 2-2](#_Toc135113810)

[2.6 Definition of Terms and Acronyms 2-3](#_Toc135113811)

[2.7 Conventions 2-6](#_Toc135113812)

[Section 3 VISA Textual Language Bindings 3-1](#_Toc135113814)

[3.1 Type Assignments 3-1](#_Toc135113815)

[3.1.1 Type Assignments for WIN95 and WINNT Frameworks 3-5](#_Toc135113816)

[3.1.2 Type Assignments for WIN64 Framework 3-5](#_Toc135113817)

[3.2 Operation Prototypes 3-6](#_Toc135113820)

[3.2.1 Operation Prototypes for WIN95 and WINNT Frameworks 3-11](#_Toc135113821)

[3.2.2 Operation Prototypes for WIN64 Framework 3-16](#_Toc135113822)

[3.3 Completion and Error Codes 3-17](#_Toc135113825)

[3.4 Attribute Values 3-21](#_Toc135113826)

[3.5 Event Type Values 3-26](#_Toc135113827)

[3.6 Values and Ranges 3-27](#_Toc135113828)

[3.7 Library Requirements 3-31](#_Toc135113829)

[3.7.1 Library Requirements for WINNT and WIN64 Frameworks 3-31](#_Toc135113830)

[3.8 Miscellaneous 3-34](#_Toc135113833)

[Appendix A Implementation Files A-1](#_Toc135113834)

[A.1 Contents of visatype.h File A-1](#_Toc135113835)

[A.2 Contents of visa.h File A-4](#_Toc135113836)

[A.3 Contents of visa32.bas File A-18](#_Toc135113837)

[A.4 Contents of visa32.def File A-28](#_Toc135113838)

[A.5 Contents of visa64.def File A-30](#_Toc135113839)

**Tables**

[Table 3.1.1. Type Assignments for VISA and Instrument Drivers 3-1](#_Toc135113888)

[Table 3.1.2. Type Assignments for VISA Only 3-4](#_Toc135113889)

[Table 3.2.1. ANSI C Bindings for VISA Operations 3-6](#_Toc135113890)

[Table 3.2.2. Visual Basic Bindings for VISA Operations for the WIN95 and WINNT Frameworks 3-12](#_Toc135113891)

[Table 3.3.1. Completion and Error Codes 3-17](#_Toc135113892)

[Table 3.4.1. Attribute Values 3-21](#_Toc135113893)

[Table 3.5.1. Event Type Values 3-26](#_Toc135113894)

[Table 3.6.1. Values and Ranges 3-27](#_Toc135113895)

[Table 3.7.1. Procedure Definition Exports for the WINNT and WIN64 Frameworks 3-31](#_Toc135113896)

[Table 3.8.1. Bit Pattern for Attributes 3-36](#_Toc135113897)

[Table 3.8.2. Bit Pattern for Status Codes 3-36](#_Toc135113898)

# Introduction to the VXI*plug&play* Systems Alliance and the IVI Foundation

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

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

# Overview of VISA Implementation Specification

This section introduces the VISA Implementation Specification for Textual Languages. This specification is a document authored by the VXI*plug&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

## 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. VXI*plug&play* drivers written to the VISA specifications can execute on VXI*plug&play* system frameworks that have the VISA I/O library.

The VISA specification provides a common standard for the VXI*plug&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.*

## 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 VXI*plug&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.

## 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.

## 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).

## 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, VXI*plug&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

## 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 *Session*. 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 *class* 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 *object* in object-oriented architectures. For VISA, *resource* more specifically refers to a particular implementation (or *instance* in object-oriented terms) of a Resource Class. In VISA, every defined software module is a resource. |
| **Session** | The same as *Communication Channel*. 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 VXI*plug&play* 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. |
|  |  |

## 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.

# VISA Textual Language Bindings

## 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. |
| ViVAList | 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.

### 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.

### 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.

## 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);

(continues)

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

ViStatus viReadToFile (ViSession vi, ViConstString filename, ViUInt32 cnt, ViPUInt32 retCnt);

ViStatus viWrite (ViSession vi, ViBuf buf, ViUInt32 cnt, ViPUInt32 retCnt);

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 parms);

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 parms);

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);

(continues)

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

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);

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);

(continues)

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

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);

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);

ViStatus viPxiReserveTriggers(ViSession vi, ViInt16 cnt, ViAInt16 trigBuses, ViAInt16 trigLines, ViPInt16 failureIndex);

**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.

### 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

(continues)

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

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

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

(continues)

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

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 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)

(continues)

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

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

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.

### 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.

## 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\_LINE\_NRESERVED | BFFF0043h |
| 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\_CNTLR | 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).

## 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\_32 | 3FFF0221h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR1\_32 | 3FFF0222h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR2\_32 | 3FFF0223h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR3\_32 | 3FFF0224h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR4\_32 | 3FFF0225h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR5\_32 | 3FFF0226h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0\_32 | 3FFF0231h |

(continues)

Table 3.4.1. Attribute Values (Continued)

|  |  |
| --- | --- |
| **Attribute Names** | **Values** |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1\_32 | 3FFF0232h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2\_32 | 3FFF0233h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3\_32 | 3FFF0234h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4\_32 | 3FFF0235h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5\_32 | 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\_MESSAGE\_KB | 3FFF0302h |
| VI\_ATTR\_TCPIP\_IS\_HISLIP | 3FFF0303h |
| VI\_ATTR\_PXI\_RECV\_INTR\_SEQ | 3FFF4240h |
| VI\_ATTR\_PXI\_RECV\_INTR\_DATA | 3FFF4241h |
| VI\_ATTR\_PXI\_SRC\_TRIG\_BUS | 3FFF020Dh |
| VI\_ATTR\_PXI\_DEST\_TRIG\_BUS | 3FFF020Eh |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR0\_64 | 3FFF0228h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR1\_64 | 3FFF0229h |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR2\_64 | 3FFF022Ah |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR3\_64 | 3FFF022Bh |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR4\_64 | 3FFF022Ch |
| VI\_ATTR\_PXI\_MEM\_BASE\_BAR5\_64 | 3FFF022Dh |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0\_64 | 3FFF0238h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1\_64 | 3FFF0239h |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2\_64 | 3FFF023Ah |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3\_64 | 3FFF023Bh |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4\_64 | 3FFF023Ch |
| VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5\_64 | 3FFF023Dh |
| VI\_ATTR\_PXI\_ALLOW\_WRITE\_COMBINE | 3FFF0246h |

**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.

## 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.

## Values and Ranges

Table 3.6.1 shows the values used in all framework bindings.

Table 3.6.1. Values and Ranges

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Value** |  | **Name** | **Value** |
| VI\_FIND\_BUFLEN | 256 |  | VI\_TRIG\_TTL5 | 5 |
| VI\_NULL | 0 |  | VI\_TRIG\_TTL6 | 6 |
| VI\_TRUE | 1 |  | VI\_TRIG\_TTL7 | 7 |
| VI\_FALSE | 0 |  | VI\_TRIG\_ECL0 | 8 |
| VI\_INTF\_GPIB | 1 |  | VI\_TRIG\_ECL1 | 9 |
| VI\_INTF\_VXI | 2 |  | VI\_TRIG\_PANEL\_IN | 27 |
| VI\_INTF\_GPIB\_VXI | 3 |  | VI\_TRIG\_PANEL\_OUT | 28 |
| VI\_INTF\_ASRL | 4 |  | VI\_TRIG\_PROT\_DEFAULT | 0 |
| VI\_INTF\_TCPIP | 6 |  | VI\_TRIG\_PROT\_ON | 1 |
| VI\_NORMAL | 1 |  | VI\_TRIG\_PROT\_OFF | 2 |
| VI\_FDC | 2 |  | VI\_TRIG\_PROT\_SYNC | 5 |
| VI\_HS488 | 3 |  | VI\_READ\_BUF | 1 |
| VI\_ASRL488 | 4 |  | VI\_WRITE\_BUF | 2 |
| VI\_FDC\_NORMAL | 1 |  | VI\_READ\_BUF\_DISCARD | 4 |
| VI\_FDC\_STREAM | 2 |  | VI\_WRITE\_BUF\_DISCARD | 8 |
| VI\_A16\_SPACE | 1 |  | VI\_ASRL\_IN\_BUF | 16 |
| VI\_A24\_SPACE | 2 |  | VI\_ASRL\_OUT\_BUF | 32 |
| VI\_A32\_SPACE | 3 |  | VI\_ASRL\_IN\_BUF\_DISCARD | 64 |
| VI\_UNKNOWN\_SLOT | -1 |  | VI\_ASRL\_OUT\_BUF\_DISCARD | 128 |
| VI\_UNKNOWN\_LA | -1 |  | VI\_FLUSH\_ON\_ACCESS | 1 |
| VI\_UNKNOWN\_LEVEL | -1 |  | VI\_FLUSH\_WHEN\_FULL | 2 |
| VI\_QUEUE | 1 |  | VI\_FLUSH\_DISABLE | 3 |
| VI\_HNDLR | 2 |  | VI\_NMAPPED | 1 |
| VI\_SUSPEND\_HNDLR | 4 |  | VI\_USE\_OPERS | 2 |
| VI\_ALL\_MECH | FFFFh |  | VI\_DEREF\_ADDR | 3 |
| VI\_ANY\_HNDLR | 0 |  | VI\_TMO\_IMMEDIATE | 0 |
| VI\_TRIG\_SW | -1 |  | VI\_TMO\_INFINITE | FFFFFFFFh |
| VI\_TRIG\_TTL0 | 0 |  | VI\_NO\_LOCK | 0 |
| VI\_TRIG\_TTL1 | 1 |  | VI\_EXCLUSIVE\_LOCK | 1 |
| VI\_TRIG\_TTL2 | 2 |  | VI\_SHARED\_LOCK | 2 |
| VI\_TRIG\_TTL3 | 3 |  | VI\_LOAD\_CONFIG | 4 |
| VI\_TRIG\_TTL4 | 4 |  | VI\_NO\_SEC\_ADDR | FFFFh |

(continues)

Table 3.6.1. Values and Ranges (Continued)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Value** |  | **Name** | **Value** |
| VI\_ASRL\_PAR\_NONE | 0 |  | VI\_DATA\_PRIV | 0 |
| VI\_ASRL\_PAR\_ODD | 1 |  | VI\_DATA\_NPRIV | 1 |
| VI\_ASRL\_PAR\_EVEN | 2 |  | VI\_PROG\_PRIV | 2 |
| VI\_ASRL\_PAR\_MARK | 3 |  | VI\_PROG\_NPRIV | 3 |
| VI\_ASRL\_PAR\_SPACE | 4 |  | VI\_BLCK\_PRIV | 4 |
| VI\_ASRL\_STOP\_ONE | 10 |  | VI\_BLCK\_NPRIV | 5 |
| VI\_ASRL\_STOP\_ONE5 | 15 |  | VI\_D64\_PRIV | 6 |
| VI\_ASRL\_STOP\_TWO | 20 |  | VI\_D64\_NPRIV | 7 |
| VI\_ASRL\_FLOW\_NONE | 0 |  | VI\_LOCAL\_SPACE | 0 |
| VI\_ASRL\_FLOW\_XON\_XOFF | 1 |  | VI\_GPIB\_REN\_DEASSERT | 0 |
| VI\_ASRL\_FLOW\_RTS\_CTS | 2 |  | VI\_GPIB\_REN\_ASSERT | 1 |
| VI\_ASRL\_FLOW\_DTR\_DSR | 4 |  | VI\_GPIB\_REN\_DEASSERT\_GTL | 2 |
| VI\_ASRL\_END\_NONE | 0 |  | VI\_GPIB\_REN\_ASSERT\_ADDRESS | 3 |
| VI\_ASRL\_END\_LAST\_BIT | 1 |  | VI\_GPIB\_REN\_ASSERT\_LLO | 4 |
| VI\_ASRL\_END\_TERMCHAR | 2 |  | VI\_GPIB\_REN\_ASSERT\_ADDRESS\_LLO | 5 |
| VI\_ASRL\_END\_BREAK | 3 |  | VI\_GPIB\_REN\_ADDRESS\_GTL | 6 |
| VI\_BIG\_ENDIAN | 0 |  | VI\_VXI\_CMD16 | 0200h |
| VI\_LITTLE\_ENDIAN | 1 |  | VI\_VXI\_CMD16\_RESP16 | 0202h |
| VI\_WIDTH\_8 | 1 |  | VI\_VXI\_RESP16 | 0002h |
| VI\_WIDTH\_16 | 2 |  | VI\_VXI\_CMD32 | 0400h |
| VI\_WIDTH\_32 | 4 |  | VI\_VXI\_CMD32\_RESP16 | 0402h |
| VI\_STATE\_ASSERTED | 1 |  | VI\_VXI\_CMD32\_RESP32 | 0404h |
| VI\_STATE\_UNASSERTED | 0 |  | VI\_VXI\_RESP32 | 0004h |
| VI\_STATE\_UNKNOWN | -1 |  | VI\_GPIB\_ATN\_DEASSERT | 0 |
| VI\_GPIB\_HS488\_DISABLED | 0 |  | VI\_GPIB\_ATN\_ASSERT | 1 |
| VI\_GPIB\_HS488\_NIMPL | -1 |  | VI\_GPIB\_ATN\_DEASSERT\_HANDSHAKE | 2 |
| VI\_VXI\_CLASS\_MEMORY | 0 |  | VI\_GPIB\_ATN\_ASSERT\_IMMEDIATE | 3 |
| VI\_VXI\_CLASS\_EXTENDED | 1 |  | VI\_ASSERT\_SIGNAL | -1 |
| VI\_VXI\_CLASS\_MESSAGE | 2 |  | VI\_ASSERT\_USE\_ASSIGNED | 0 |
| VI\_VXI\_CLASS\_REGISTER | 3 |  | VI\_ASSERT\_IRQ1 | 1 |
| VI\_VXI\_CLASS\_OTHER | 4 |  | VI\_ASSERT\_IRQ2 | 2 |
| VI\_UTIL\_ASSERT\_SYSRESET | 1 |  | VI\_ASSERT\_IRQ3 | 3 |
| VI\_UTIL\_ASSERT\_SYSFAIL | 2 |  | VI\_ASSERT\_IRQ4 | 4 |
| VI\_UTIL\_DEASSERT\_SYSFAIL | 3 |  | VI\_ASSERT\_IRQ5 | 5 |
| VI\_TRIG\_ALL | -2 |  | VI\_ASSERT\_IRQ6 | 6 |

(continues)

Table 3.6.1. Values and Ranges (Continued)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Value** |  | **Name** | **Value** |
| VI\_ASSERT\_IRQ7 | 7 |  | VI\_IO\_IN\_BUF | 16 |
| VI\_GPIB\_UNADDRESSED | 0 |  | VI\_IO\_OUT\_BUF | 32 |
| VI\_GPIB\_TALKER | 1 |  | VI\_IO\_IN\_BUF\_DISCARD | 64 |
| VI\_GPIB\_LISTENER | 2 |  | VI\_IO\_OUT\_BUF\_DISCARD | 128 |
| VI\_INTF\_USB | 7 |  | VI\_PROT\_NORMAL | 1 |
| VI\_PROT\_FDC | 2 |  | VI\_PROT\_HS488 | 3 |
| VI\_PROT\_4882\_STRS | 4 |  | VI\_PROT\_USBTMC\_VENDOR | 5 |
| VI\_OPAQUE\_SPACE | FFFFh |  | VI\_UNKNOWN\_CHASSIS | -1 |
| VI\_INTF\_PXI | 5 |  | VI\_UNKNOWN\_TRIG | -1 |
| VI\_PXI\_ALLOC\_SPACE | 9 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_0 | 1000 |
| VI\_PXI\_CFG\_SPACE | 10 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_1 | 1001 |
| VI\_PXI\_BAR0\_SPACE | 11 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_2 | 1002 |
| VI\_PXI\_BAR1\_SPACE | 12 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_3 | 1003 |
| VI\_PXI\_BAR2\_SPACE | 13 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_4 | 1004 |
| VI\_PXI\_BAR3\_SPACE | 14 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_5 | 1005 |
| VI\_PXI\_BAR4\_SPACE | 15 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_6 | 1006 |
| VI\_PXI\_BAR5\_SPACE | 16 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_7 | 1007 |
| VI\_PXI\_ADDR\_NONE | 0 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_8 | 1008 |
| VI\_PXI\_ADDR\_MEM | 1 |  | VI\_PXI\_LBUS\_STAR\_TRIG\_BUS\_9 | 1009 |
| VI\_PXI\_ADDR\_IO | 2 |  | VI\_PXI\_STAR\_TRIG\_CONTROLLER | 1413 |
| VI\_PXI\_ADDR\_CFG | 3 |  | VI\_TRIG\_PROT\_RESERVE | 6 |
| VI\_A64\_SPACE | 4 |  | VI\_TRIG\_PROT\_UNRESERVE | 7 |
| VI\_WIDTH\_64 | 8 |  | VI\_TRIG\_ECL2 | 10 |
| VI\_TRIG\_ECL3 | 11 |  | VI\_TRIG\_ECL4 | 12 |
| VI\_TRIG\_ECL5 | 13 |  | VI\_TRIG\_STAR\_SLOT1 | 14 |
| VI\_TRIG\_STAR\_SLOT2 | 15 |  | VI\_TRIG\_STAR\_SLOT3 | 16 |
| VI\_TRIG\_STAR\_SLOT4 | 17 |  | VI\_TRIG\_STAR\_SLOT5 | 18 |
| VI\_TRIG\_STAR\_SLOT6 | 19 |  | VI\_TRIG\_STAR\_SLOT7 | 20 |
| VI\_TRIG\_STAR\_SLOT8 | 21 |  | VI\_TRIG\_STAR\_SLOT9 | 22 |
| VI\_TRIG\_STAR\_SLOT10 | 23 |  | VI\_TRIG\_STAR\_SLOT11 | 24 |
| VI\_TRIG\_STAR\_SLOT12 | 25 |  | VI\_TRIG\_STAR\_INSTR | 26 |
| VI\_TRIG\_STAR\_VXI0 | 29 |  | VI\_TRIG\_STAR\_VXI1 | 30 |
| VI\_TRIG\_STAR\_VXI2 | 31 |  | VI\_D64\_2EVME | 8 |
| VI\_D64\_SST160 | 9 |  | VI\_D64\_SST267 | 10 |
| VI\_D64\_SST320 | 11 |  |  |  |

Table 3.6.1. Values and Ranges (Continued)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Value** |  | **Name** | **Value** |
| VI\_TRIG\_TTL8 | 32 |  | VI\_TRIG\_TTL10 | 34 |
| VI\_TRIG\_TTL9 | 33 |  | VI\_TRIG\_TTL11 | 35 |

**RULE 3.6.1**

All values and ranges specified in Table 3.6.1 **SHALL** appear in the visa.h file.

**RULE 3.6.2**

All values and ranges specified in Table 3.6.1 as of version 5.0 of this specification **SHALL** appear in the visa32.bas file.

**OBSERVATION 3.6.1**

The level of Visual Basic support has been frozen as of the 5.0 specification.

**RULE 3.6.3**

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 3.6.4**

The range of the attribute VI\_ATTR\_USER\_DATA **SHALL** be 0 to FFFFFFFFh.

## 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.

### 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** |  | **Entry Point** | **Ordinal Number** |
| viGetDefaultRM | 128 |  | viClose | 132 |
| viOpenDefaultRM | 141 |  | viGetAttribute | 133 |
| viFindRsrc | 129 |  | viSetAttribute | 134 |
| viFindNext | 130 |  | viStatusDesc | 142 |
| viOpen | 131 |  | viTerminate | 143 |
| viLock | 144 |  | viReadSTB | 259 |
| viUnlock | 145 |  | viClear | 260 |
| viEnableEvent | 135 |  | viSetBuf | 267 |
| viDisableEvent | 136 |  | viFlush | 268 |
| viDiscardEvents | 137 |  | viPrintf | 269 |
| viWaitOnEvent | 138 |  | viVPrintf | 270 |
| viInstallHandler | 139 |  | viScanf | 271 |
| viUninstallHandler | 140 |  | viVScanf | 272 |
| viMove | 200 |  | viQueryf | 279 |
| viMoveAsync | 201 |  | viVQueryf | 280 |
| viBufWrite | 202 |  | viIn8 | 273 |
| viBufRead | 203 |  | viOut8 | 274 |
| viSPrintf | 204 |  | viIn16 | 261 |
| viVSPrintf | 205 |  | viOut16 | 262 |
| viSScanf | 206 |  | viIn32 | 281 |
| viVSScanf | 207 |  | viOut32 | 282 |
| viGpibControlREN | 208 |  | viMoveIn8 | 283 |
| viVxiCommandQuery | 209 |  | viMoveOut8 | 284 |

(continues)

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entry Point** | **Ordinal Number** |  | **Entry Point** | **Ordinal Number** |
| viRead | 256 |  | viMoveIn16 | 285 |
| viReadAsync | 277 |  | viMoveOut16 | 286 |
| viWrite | 257 |  | viMoveIn32 | 287 |
| viWriteAsync | 278 |  | viMoveOut32 | 288 |
| viAssertTrigger | 258 |  | viPeek8 | 275 |
| viMapAddress | 263 |  | viPoke8 | 276 |
| viUnmapAddress | 264 |  | viPeek16 | 265 |
| viMemAlloc | 291 |  | viPoke16 | 266 |
| viMemFree | 292 |  | viPeek32 | 289 |
| viGpibControlATN | 210 |  | viPoke32 | 290 |
| viGpibSendIFC | 211 |  | viParseRsrc | 146 |
| viGpibCommand | 212 |  | viMapTrigger | 216 |
| viGpibPassControl | 213 |  | viUnmapTrigger | 217 |
| viAssertUtilSignal | 214 |  | viWriteFromFile | 218 |
| viAssertIntrSignal | 215 |  | viReadToFile | 219 |
| viParseRsrcEx | 147 |  | viUsbControlOut | 293 |
| viUsbControlIn | 294 |  | 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 |
| viPxiReserveTriggers | 295 |  |  |  |

**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.

## Miscellaneous

**RULE 3.8.1**

Every VISA 5.7 implementation **SHALL** provide the following #define in the visa.h file:  
  
#define VI\_SPEC\_VERSION (0x00500700UL)

**RULE 3.8.2**

Every VISA 5.0 or higher implementation **SHALL** provide the following constant in the visa32.bas file:  
  
Global Const VI\_SPEC\_VERSION = &H00500000&

**OBSERVATION 3.8.1**

The level of Visual Basic support has been frozen as of the 5.0 specification.

**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 3.8.2**

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 VXI*plug&play* applications and instrument drivers compiled without user-defined macros.

**OBSERVATION 3.8.3**

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 3.8.4**

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 3.8.5**

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 3.8.6**

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. It is provided as a reference and may not have the same date or version as the actual file installed on the system.

/\*---------------------------------------------------------------------------\*/

/\* Distributed by IVI Foundation Inc. \*/

/\* \*/

/\* Do not modify the contents of this file. \*/

/\*---------------------------------------------------------------------------\*/

/\* \*/

/\* Title : VISATYPE.H \*/

/\* Date : 02-01-2016 \*/

/\* 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\_FUNCC \_\_fastcall

#define \_VI\_FUNCH \_\_fastcall

#define \_VI\_SIGNED signed

#elif (defined(WIN32) || defined(\_WIN32) || defined(\_\_WIN32\_\_) || defined(\_\_NT\_\_)) && !defined(\_NI\_mswin16\_) #define \_VI\_FAR

#define \_VI\_FUNC \_\_stdcall

#define \_VI\_FUNCC \_\_cdecl

#define \_VI\_FUNCH \_\_stdcall

#define \_VI\_SIGNED signed

#elif defined(\_CVI\_) && defined(\_NI\_i386\_)

#define \_VI\_FAR

#define \_VI\_FUNC \_pascal

#define \_VI\_FUNCC

#define \_VI\_FUNCH \_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\_FUNCC \_far \_cdecl \_export

#define \_VI\_FUNCH \_far \_pascal

#define \_VI\_SIGNED signed

#elif (defined(hpux) || defined(\_\_hpux)) && (defined(\_\_cplusplus) || defined(\_\_cplusplus\_\_))

#define \_VI\_FAR

#define \_VI\_FUNC

#define \_VI\_FUNCC

#define \_VI\_FUNCH

#define \_VI\_SIGNED

#else

#define \_VI\_FAR

#define \_VI\_FUNC

#define \_VI\_FUNCC

#define \_VI\_FUNCH

#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)) || defined(\_\_LCC\_\_) || (defined(\_\_GNUC\_\_) && (\_\_GNUC\_\_ >= 3)) || (defined(\_\_clang\_\_) && (\_\_clang\_major\_\_ >= 3))

typedef unsigned \_\_int64 ViUInt64;

typedef \_VI\_SIGNED \_\_int64 ViInt64;

#define \_VI\_INT64\_UINT64\_DEFINED

#if defined(\_WIN64)

#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. It is provided as a reference and may not have the same date or version as the actual file installed on the system.

/\*---------------------------------------------------------------------------\*/

/\* Distributed by IVI Foundation Inc. \*/

/\* \*/

/\* Do not modify the contents of this file. \*/

/\*---------------------------------------------------------------------------\*/

/\* \*/

/\* Title : VISA.H \*/

/\* Date : 02-01-2016 \*/

/\* Purpose : Include file for the VISA Library 5.7 specification \*/

/\* \*/

/\*---------------------------------------------------------------------------\*/

#ifndef \_\_VISA\_HEADER\_\_

#define \_\_VISA\_HEADER\_\_

#include <stdarg.h>

#if !defined(\_\_VISATYPE\_HEADER\_\_)

#include "visatype.h"

#endif

#define VI\_SPEC\_VERSION (0x00500700UL)

#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\_FUNCH \_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,

ViPUInt32 retCnt, ViChar \_VI\_FAR desc[]);

ViStatus \_VI\_FUNC viFindNext (ViFindList vi, ViChar \_VI\_FAR desc[]);

ViStatus \_VI\_FUNC viParseRsrc (ViSession rmSesn, ViRsrc rsrcName,

ViPUInt16 intfType, ViPUInt16 intfNum);

ViStatus \_VI\_FUNC viParseRsrcEx (ViSession rmSesn, ViRsrc rsrcName, ViPUInt16 intfType,

ViPUInt16 intfNum, ViChar \_VI\_FAR rsrcClass[],

ViChar \_VI\_FAR expandedUnaliasedName[],

ViChar \_VI\_FAR aliasIfExists[]);

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,

ViKeyId requestedKey, ViChar \_VI\_FAR accessKey[]);

ViStatus \_VI\_FUNC viUnlock (ViSession vi);

ViStatus \_VI\_FUNC viEnableEvent (ViSession vi, ViEventType eventType, ViUInt16 mechanism,

ViEventFilter context);

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,

ViPEventType outEventType, ViPEvent outContext);

ViStatus \_VI\_FUNC viInstallHandler(ViSession vi, ViEventType eventType, ViHndlr handler,

ViAddr userHandle);

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,

ViPUInt32 retCnt);

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,

ViPUInt32 retCnt);

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\_FUNCC viPrintf (ViSession vi, ViString writeFmt, ...);

ViStatus \_VI\_FUNC viVPrintf (ViSession vi, ViString writeFmt, ViVAList params);

ViStatus \_VI\_FUNCC viSPrintf (ViSession vi, ViPBuf buf, ViString writeFmt, ...);

ViStatus \_VI\_FUNC viVSPrintf (ViSession vi, ViPBuf buf, ViString writeFmt,

ViVAList parms);

ViStatus \_VI\_FUNCC viScanf (ViSession vi, ViString readFmt, ...);

ViStatus \_VI\_FUNC viVScanf (ViSession vi, ViString readFmt, ViVAList params);

ViStatus \_VI\_FUNCC viSScanf (ViSession vi, ViBuf buf, ViString readFmt, ...);

ViStatus \_VI\_FUNC viVSScanf (ViSession vi, ViBuf buf, ViString readFmt,

ViVAList parms);

ViStatus \_VI\_FUNCC viQueryf (ViSession vi, ViString writeFmt, ViString readFmt, ...);

ViStatus \_VI\_FUNC viVQueryf (ViSession vi, ViString writeFmt, ViString readFmt,

ViVAList params);

/\*- 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,

ViUInt16 srcWidth, ViUInt16 destSpace,

ViBusAddress destOffset, ViUInt16 destWidth,

ViBusSize srcLength);

ViStatus \_VI\_FUNC viMoveAsync (ViSession vi, ViUInt16 srcSpace, ViBusAddress srcOffset,

ViUInt16 srcWidth, ViUInt16 destSpace,

ViBusAddress destOffset, ViUInt16 destWidth,

ViBusSize srcLength, ViPJobId jobId);

#if defined(\_VI\_INT64\_UINT64\_DEFINED)

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);

ViStatus \_VI\_FUNC viPxiReserveTriggers(ViSession vi, ViInt16 cnt, ViAInt16 trigBuses,

ViAInt16 trigLines, ViPInt16 failureIndex);

/\*- 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\_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\_KEEPALIVE (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\_SRC\_TRIG\_BUS (0x3FFF020DUL)

#define VI\_ATTR\_PXI\_DEST\_TRIG\_BUS (0x3FFF020EUL)

#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\_32 (0x3FFF0221UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR1\_32 (0x3FFF0222UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR2\_32 (0x3FFF0223UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR3\_32 (0x3FFF0224UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR4\_32 (0x3FFF0225UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR5\_32 (0x3FFF0226UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR0\_64 (0x3FFF0228UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR1\_64 (0x3FFF0229UL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR2\_64 (0x3FFF022AUL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR3\_64 (0x3FFF022BUL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR4\_64 (0x3FFF022CUL)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR5\_64 (0x3FFF022DUL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0\_32 (0x3FFF0231UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1\_32 (0x3FFF0232UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2\_32 (0x3FFF0233UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3\_32 (0x3FFF0234UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4\_32 (0x3FFF0235UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5\_32 (0x3FFF0236UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0\_64 (0x3FFF0238UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1\_64 (0x3FFF0239UL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2\_64 (0x3FFF023AUL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3\_64 (0x3FFF023BUL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4\_64 (0x3FFF023CUL)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5\_64 (0x3FFF023DUL)

#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\_PXI\_ALLOW\_WRITE\_COMBINE (0x3FFF0246UL)

#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\_TCPIP\_IS\_HISLIP (0x3FFF0303UL)

#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)

#define VI\_ATTR\_PXI\_RECV\_INTR\_SEQ (0x3FFF4240UL)

#define VI\_ATTR\_PXI\_RECV\_INTR\_DATA (0x3FFF4241UL)

/\*- 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)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR0 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR0\_64)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR1 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR1\_64)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR2 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR2\_64)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR3 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR3\_64)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR4 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR4\_64)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR5 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR5\_64)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0\_64)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1\_64)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2\_64)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3\_64)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4\_64)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5\_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)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR0 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR0\_32)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR1 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR1\_32)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR2 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR2\_32)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR3 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR3\_32)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR4 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR4\_32)

#define VI\_ATTR\_PXI\_MEM\_BASE\_BAR5 (VI\_ATTR\_PXI\_MEM\_BASE\_BAR5\_32)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR0\_32)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR1\_32)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR2\_32)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR3\_32)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR4\_32)

#define VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5 (VI\_ATTR\_PXI\_MEM\_SIZE\_BAR5\_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 (0x3FFF007DL) /\* 3FFF007D, 1073676413 \*/

#define VI\_SUCCESS\_TRIG\_MAPPED (0x3FFF007EL) /\* 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 (0x3FFF0077L) /\* 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\_NSUP\_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\_LINE\_NRESERVED (\_VI\_ERROR+0x3FFF0043L) /\* BFFF0043, -1073807293 \*/

#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\_CNTLR (\_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) & 0xFFF00000UL) >> 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\_ECL2 (10)

#define VI\_TRIG\_ECL3 (11)

#define VI\_TRIG\_ECL4 (12)

#define VI\_TRIG\_ECL5 (13)

#define VI\_TRIG\_STAR\_SLOT1 (14)

#define VI\_TRIG\_STAR\_SLOT2 (15)

#define VI\_TRIG\_STAR\_SLOT3 (16)

#define VI\_TRIG\_STAR\_SLOT4 (17)

#define VI\_TRIG\_STAR\_SLOT5 (18)

#define VI\_TRIG\_STAR\_SLOT6 (19)

#define VI\_TRIG\_STAR\_SLOT7 (20)

#define VI\_TRIG\_STAR\_SLOT8 (21)

#define VI\_TRIG\_STAR\_SLOT9 (22)

#define VI\_TRIG\_STAR\_SLOT10 (23)

#define VI\_TRIG\_STAR\_SLOT11 (24)

#define VI\_TRIG\_STAR\_SLOT12 (25)

#define VI\_TRIG\_STAR\_INSTR (26)

#define VI\_TRIG\_PANEL\_IN (27)

#define VI\_TRIG\_PANEL\_OUT (28)

#define VI\_TRIG\_STAR\_VXI0 (29)

#define VI\_TRIG\_STAR\_VXI1 (30)

#define VI\_TRIG\_STAR\_VXI2 (31)

#define VI\_TRIG\_TTL8 (32)

#define VI\_TRIG\_TTL9 (33)

#define VI\_TRIG\_TTL10 (34)

#define VI\_TRIG\_TTL11 (35)

#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 (0xFFFFFFFFUL)

#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\_D64\_2EVME (8)

#define VI\_D64\_SST160 (9)

#define VI\_D64\_SST267 (10)

#define VI\_D64\_SST320 (11)

#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\_RESP16 (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. It is provided as a reference and may not have the same date or version as the actual file installed on the system.

' -------------------------------------------------------------------------

' 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 = &H00500000&

' - 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\_CNTLR = &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 = &HFFFFFFFF&

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 = &HBFFF000E&

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

viPxiReserveTriggers @295

## 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

viPxiReserveTriggers @295