Tektronix®

TekVISA
Reference Manual
Version 3.0

Register now!
Click the following link to protect your product.
tek.com/register

071-1104-03 December 2023

Copyright © 2023, Tektronix. 2023 All rights reserved. Licensed software products are owned by Tektronix or its subsidiaries or suppliers, and are protected by national copyright laws and international treaty provisions. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specifications and price change privileges reserved. All other trade names referenced are the service marks, trademarks, or registered trademarks of their respective companies.

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
US

For product information, sales, service, and technical support visit *tek.com* to find contacts in your area. For warranty information visit *tek.com/warranty*.

Contents

Resource manager functions and operations

Open a resource manager session	`viOpenDefaultRM (ViPSession sesn)`
Find the first of possibly many instruments	`viFindRsrc(ViSession sesn, ViConstString expr, ViPFindList findlist, ViPUInt32 retCount, ViPRsrc instrdesc)`
Find the next instrument in a list of instruments	`viFindNext(ViFindList findlist, ViPRsrc instrdesc)`
Open an instrument session	`viOpen(ViSession sesn, ViConstRsrc rsrcName, ViAccessMode accessmode, ViUInt32 timeout, ViPSession vi)`
Parse a resource string to get the interface information	`viParseRsrc(ViSession sesn, ViConstRsrc rsrcName, ViUint16 intfType, ViUInt intfNum)`

Resource template operations

Close a session (instrument, event, find list, or resource manager)	`viClose(ViObject vi)`
Set an attribute; see attributes	`viSetAttribute(ViObject vi, ViAttr attribute, ViAttrState attrState)`
Get the current value of an attribute	`viGetAttribute(ViObject vi, ViAttr attribute, ViPAttrState attrState)`
Convert a status result to a text string	`viStatusDesc(ViObject vi, ViStatus status, ViPString desc)`
Terminate an asynchronous operation	`viTerminate(ViObject vi, ViUInt16 degree, ViJobId jobId)`
Control the access to an instrument	`viLock(ViSession vi, ViAccessMode lockType, ViUInt32 timeout, ViConstKeyId requestedKey, ViPKeyId accessKey)`
Allow others to access an instrument	`viUnlock(ViSession vi)`
Prototype for callback handler to be called when a particular event occurs	`viEventHandler(ViSession vi, ViEventType eventType, ViEvent context, ViAddr userHandle)`
Allow an event to be reported	`viEnableEvent(ViSession vi, ViEventType eventType, ViUInt16 mechanism, ViEventFilter context)`
Prevent events from being reported	`viDisableEvent(ViSession vi, ViEventType eventType, ViUInt16 mechanism)`
Discard all pending occurrences of an event	`viDiscardEvents(ViSession vi, ViEventType eventType, ViUInt16 mechanism)`
Wait for an event to occur	`viWaitOnEvent(ViSession vi, ViEventType inEventType, ViUInt32 timeout, ViPEventType outEventType, ViPEvent outContext)`
Register an event handler	`viInstallHandler(ViSession vi, ViEventType eventType, ViHndlr handler, ViAddr userHandle)`
Remove an event handler	`viUninstallHandler(ViSession vi, ViEventType eventType, ViHndlr handler, ViAddr userHandle)`

Basic I/O operations

Read from an instrument	`viRead(ViSession vi, ViPBuf buf, ViUInt32 count, ViPUInt32 retCount)`
Read from an instrument but run while reading	`viReadAsync(ViSession vi, ViPBuf buf, ViUInt32 count, ViPJobId jobId)`
Write to an instrument	`viWrite(ViSession vi, ViConstBuf buf, ViUInt32 count, ViPUInt32 retCount)`
Write to an instrument but run while writing	`viWriteAsync(ViSession vi, ViConstBuf buf, ViUInt32 count, ViPJobId jobId)`
Generate a hardware or software trigger	`viAssertTrigger(ViSession vi, ViUInt16 protocol)`
Read the status byte	`viReadSTB(ViSession vi, ViPUInt16 status)`
Send a bus-dependent clear command	`viClear(ViSession vi)`
Read data synchronously from a device, and stores the transferred data in a file	`viReadToFile(ViSession vi, ViString filename, ViUInt32 count, ViUInt32 retCount)`
Take data from a file and write it to a device synchronously	`viWriteFromFile(ViSession vi, ViString filename, ViUInt32 count, ViUInt32 retCount)`

Formatted I/O operations

Read data synchronously from a device into the formatted I/O buffer	`viBufRead(ViSession vi, ViPBuf buf, ViUInt32 count, ViPUInt32 retCount)`
Write data synchronously to a device from the formatted I/O buffer	`viBufWrite(ViSession vi, ViBuf buf, ViUInt32 count, ViPUInt32 retCount)`
Set the size of the formatted I/O and serial I/O buffers	`viSetBuf(ViSession vi, ViUInt16 mask, ViUInt32 size)`
Empty a formatted I/O or serial I/O buffer	`viFlush(ViSession vi, ViUInt16 mask)`
Create a formatted string and send it to an instrument	`viPrintf(ViSession vi, ViConstString writeFmt,…)`
Create a formatted string and send it to an instrument using a user-supplied buffer	`viSPrintf(ViSession vi, ViPBuf buf, ViConstString writeFmt,…)`
Create a formatted string and send it to an instrument using a pointer	`viVPrintf(ViSession vi, ViConstString writeFmt, ViVAList params)`
Create a formatted string and send it to an instrument using a pointer and a user-supplied buffer	`viVSPrintf(ViSession vi, ViPBuf buf, ViConstString writeFmt,…)`
Read and extract data from an instrument, and perform formatted input	`viScanf(ViSession vi, ViConstString readFmt, …)`
Read and extract data from an instrument, and perform formatted input using a user-supplied buffer	`viSScanf(ViSession vi, ViConstBuf buf, ViConstString readFmt,…)`
Read and extract data from an instrument, and perform formatted input using a pointer	`viVScanf(ViSession vi, ViConstString readFmt, ViVAList params)`
Read and extract data from an instrument, and perform formatted input using a user-supplied buffer	`viVSScanf(ViSession vi, ViConstBuf buf, ViConstString readFmt,…)`
Write formatted data to and read formatted data from an instrument	`viQueryf(ViSession vi, ViConstString writeFmt, ViConstString readFmt,…)`
Write formatted data to and read formatted data from an instrument using a pointer	`viVQueryf(ViSession vi, ViConstString writeFmt, ViConstString readFmt, ViVAList params);`

Attribute	Type	R/W
VI_ATTR_ASRL_AVAIL_NUM	ViUInt32	RO

Table continued…

Attribute	Type	R/W
VI_ATTR_ASRL_BAUD	ViUInt32	R/W
VI_ATTR_ASRL_CTS_STATE	ViInt16	RO
VI_ATTR_4882_COMPLIANT	ViBoolean	RO
VI_ATTR_ASRL_DATA_BITS	ViUInt16	R/W
VI_ATTR_ASRL_DCD	ViInt16	RO
VI_ATTR_ASRL_DCD_STATE	ViInt16	RO
VI_ATTR_ASRL_DSR_STATE	ViInt16	RO
VI_ATTR_ASRL_DTR_STATE	ViInt16	R/W
VI_ATTR_ASRL_END_IN	ViUInt16	R/W
VI_ATTR_ASRL_END_OUT	ViUInt16	R/W
VI_ATTR_ASRL_FLOW_CNTRL	ViUInt16	R/W
VI_ATTR_FDC_CHNL	ViUInt16	R/W
VI_ATTR_ASRL_PARITY	ViUInt16	R/W
VI_ATTR_ASRL_REPLACE_CHAR	ViUInt8	R/W
VI_ATTR_ASRL_RI_STATE	ViInt16	RO
VI_ATTR_ASRL_RTS_STATE	ViInt16	R/W
VI_ATTR_ASRL_STOP_BITS	ViUInt16	R/W
VI_ATTR_ASRL_XOFF_CHAR	ViUInt8	R/W
VI_ATTR_ASRL_XON_CHAR	ViUInt8	R/W
VI_ATTR_BUFFER	ViBuf	RO
VI_ATTR_CMDR_LA	ViInt16	RO
VI_ATTR_DEST_BYTE_ORDER	ViUInt16	R/W
VI_ATTR_DEST_ACCESS_PRIV	ViUInt16	R/W
VI_ATTR_DMA_ALLOW_EN	ViBoolean	R/W
VI_ATTR_DEST_INCREMENT	ViInt32	R/W
VI_ATTR_EVENT_TYPE	ViEventType	RO
VI_ATTR_FILE_APPEND_EN	Boolean	R/W
VI_ATTR_FDC_GEN_SIGNAL_EN	ViBoolean	R/W
VI_ATTR_FDC_MODE	ViUInt16	R/W
VI_ATTR_FDC_USE_PAIR	ViBoolean	R/W
VI_ATTR_GPIB_PRIMARY_ADDR	ViUInt16	RO
VI_ATTR_GPIB_READDR_EN	ViBoolean	R/W
VI_ATTR_GPIB_SECONDARY_ADDR	ViUInt16	RO
VI_ATTR_GPIB_UNADDR_EN	ViBoolean	R/W
VI_ATTR_GPIB_REN_STATE	ViInt16	RO
VI_ATTR_INTF_INST_NAME	ViString	RO
VI_ATTR_INTF_NUM	ViUInt16	RO

Table continued…

Attribute	Type	R/W
VI_ATTR_INTF_TYPE	ViUInt16	RO
VI_ATTR_IO_PROT	ViUInt16	R/W
VI_ATTR_IMMEDIATE_SERV	viBoolean	RO
VI_ATTR_INTF_PARENT_NUM	ViUInt16	RO
VI_ATTR_JOB_ID	ViJobID	RO
VI_ATTR_MAX_QUEUE_LENGTH	ViUInt32	R/W
VI_ATTR_MAINFRAME_LA	ViInt16	RO
VI_ATTR_MEM_BASE_32	ViUInt32	RO
VI_ATTR_MEM_BASE_64	ViBusAddress64	RO
VI_ATTR_MEM_SIZE_32	ViUInt32	RO
VI_ATTR_MEM_SIZE_64	ViBusSize64	RO
VI_ATTR_MEM_SPACE	ViUInt16	RO
VI_ATTR_MANF_ID	ViUInt16	RO
VI_ATTR_MODEL_CODE	ViUInt16	RO
VI_ATTR_MANF_NAME	ViString	RO
VI_ATTR_MODEL_NAME	ViString	RO
VI_ATTR_OPER_NAME	ViString	RO
VI_ATTR_PXI_BUS_NUM	ViUInt16	RO
VI_ATTR_PXI_DEV_NUM	ViUInt16	RO
VI_ATTR_PXI_FUNC_NUM	ViUInt16	RO
VI_ATTR_PXI_SLOTPATH	ViString	RO
VI_ATTR_PXI_SLOT_LBUS_LEFT	ViInt16	RO
VI_ATTR_PXI_SLOT_LBUS_RIGHT	ViInt16	RO
VI_ATTR_PXI_TRIG_BUS	ViInt16	RO
VI_ATTR_PXI_STAR_TRIG_BUS	ViInt16	RO
VI_ATTR_PXI_STAR_TRIG_LINE	ViInt16	RO
VI_ATTR_PXI_MEM_TYPE_BARn (where n is 0,1,2,3,4,5)	ViInt16	RO
VI_ATTR_PXI_MEM_BASE_BARn (where n is 0,1,2,3,4,5)	ViBusAddress	RO
VI_ATTR_PXI_MEM_BASE_BARn_32 (where n is 0,1,2,3,4,5)	ViUInt32	RO
VI_ATTR_PXI_MEM_BASE_BARn_64n(where n is 0,1,2,3,4,5)	ViBusAddress64	RO
VI_ATTR_PXI_MEM_SIZE_BARn_32 (where n is 0,1,2,3,4,5)	ViUInt32	RO
VI_ATTR_PXI_MEM_SIZE_BARn _64 (where n is 0,1,2,3,4,5)	ViBusSize64	RO
VI_ATTR_PXI_CHASSIS	ViInt16	RO
VI_ATTR_PXI_IS_EXPRESS	ViBoolean	RO
VI_ATTR_PXI_SLOT_LWIDTH	ViInt16	RO
VI_ATTR_PXI_MAX_LWIDTH	ViInt16	RO
VI_ATTR_PXI_ACTUAL_LWIDTH	ViInt16	RO

Table continued…

Attribute	Type	R/W
VI_ATTR_PXI_DSTAR_BUS	ViInt16	RO
VI_ATTR_PXI_DSTAR_SET	ViInt16	RO
VI_ATTR_PXI_ALLOW_WRITE_COMBINE	ViBoolean	R/W
VI_ATTR_PXI_SLOT_WIDTH	ViUInt16	RO
VI_ATTR_PXI_SLOT_OFFSET	ViUInt16	RO
VI_ATTR_RD_BUF_OPER_MODE	ViUInt16	R/W
VI_ATTR_RET_COUNT	ViUInt32	RO
VI_ATTR_RM_SESSION	ViSession	RO
VI_ATTR_RSRC_IMPL_VERSION	ViVersion	RO
VI_ATTR_RSRC_LOCK_STATE	ViAccessMode	RO
VI_ATTR_RSRC_MANF_ID	ViUInt16	RO
VI_ATTR_RSRC_MANF_NAME	ViString	RO
VI_ATTR_RSRC_NAME	ViRsrc	RO
VI_ATTR_RSRC_SPEC_VERSION	ViVersion	RO
VI_ATTR_RD_BUF_SIZE	ViUInt32	RO
VI_ATTR_SEND_END_EN	ViBoolean	R/W
VI_ATTR_STATUS	ViStatus	RO
VI_ATTR_SUPPRESS_END_EN	ViBoolean	R/W
VI_ATTR_SRC_ACCESS_PRIV	ViUInt16	R/W
VI_ATTR_SLOT	ViInt16	RO
VI_ATTR_SRC_INCREMENT	ViInt32	R/W
VI_ATTR_SRC_BYTE_ORDER	ViUInt16	R/W
VI_ATTR_TCPIP_ADDR	ViString	RO
VI_ATTR_TCPIP_HOSTNAME	ViString	RO
VI_ATTR_TCPIP_DEVICE_NAME	ViString	RO
VI_ATTR_TCPIP_IS_HISLIP	ViBoolean	RO
VI_ATTR_TCPIP_SERVER_CERT	ViString	RO
VI_ATTR_TCPIP_SERVER_CERT_SIZE	ViUInt32	RO
VI_ATTR_TCPIP_HISLIP_OVERLAP_EN	ViBoolean	R/W
VI_ATTR_TCPIP_HISLIP_VERSION	ViVersion	RO
VI_ATTR_TCPIP_HISLIP_MAX_MESSAGE_KB	ViUInt32	R/W
VI_ATTR_TCPIP_HISLIP_ENCRYPTION_EN	ViBoolean	R/W
VI_ATTR_TCPIP_SERVER_CERT_ISSUER_NAME	ViString	RO
VI_ATTR_TCPIP_SERVER_CERT_SUBJECT_NAME	ViString	RO
VI_ATTR_TCPIP_SERVER_CERT_EXPIRATION_DATE	ViString	RO
VI_ATTR_TCPIP_SASL_MECHANISM	ViString	RO
VI_ATTR_TCPIP_TLS_CIPHER_SUITE	ViString	RO

Table continued…

Attribute	Type	R/W
VI_ATTR_TCPIP_SERVER_CERT_IS_PERPETUAL	ViBoolean	RO
VI_ATTR_TERMCHAR	ViUInt8	R/W
VI_ATTR_TERMCHAR_EN	ViBoolean	R/W
VI_ATTR_TMO_VALUE	ViUInt32	R/W
VI_ATTR_TRIG_ID	ViUInt16	R/W
VI_ATTR_USER_DATA	ViAddr	R/W
VI_ATTR_USB_SERIAL_NUM	ViString	RO
VI_ATTR_USB_INTFC_NUM	ViInt16	RO
VI_ATTR_USB_MAX_INTR_SIZE	ViUInt16	RW
VI_ATTR_USB_PROTOCOL	ViInt16	RO
VI_ATTR_VXI_DEV_CLASS	ViUInt16	RO
VI_ATTR_VXI_LA	ViInt16	RO
VI_ATTR_VXI_TRIG_SUPPORT	ViUInt32	RO
VI_ATTR_WR_BUF_OPER_MODE	ViUInt16	RW
VI_ATTR_WR_BUF_SIZE	ViUInt32	RO
VI_ATTR_WIN_BYTE_ORDER	ViUInt16	RW*
VI_ATTR_WIN_ACCESS_PRIV	ViUInt16	RW
VI_ATTR_WIN_ACCESS	ViUInt16	RO
VI_ATTR_WIN_BASE_ADDR_32	ViBusAddress	RO
VI_ATTR_WIN_BASE_ADDR_64	ViBusAddress64	RO
VI_ATTR_WIN_SIZE_32	ViBusSize	RO
VI_ATTR_WIN_SIZE_64	ViBusSize64	RO

Event types

VI_EVENT_EXCEPTION

VI_EVENT_IO_COMPLETION

VI_EVENT_SERVICE_REQ

Completion and error codes

- VI_SUCCESS — The operation completed successfully.
- > VI_SUCCESS — The operation succeeded conditionally. This return condition may need to be handled. See TekVISA manual for more information.
- < VI_SUCCESS — The operation failed.

Read/Write example

```
#include <visa.h>
#include <stdio.h>
int main(int argc, char* argv[]) {
ViSession rm, vi;
ViUInt32 retCnt;
ViChar buffer[256];
viOpenDefaultRM(&rm);
viOpen(rm, “GPIB0::1::INSTR”, VI_NULL, VI_NULL, &vi);
viWrite(vi, “*idn?”, 5, &retCnt);
viRead(vi, buffer, 256, &retCnt);
printf(“device: %s\n”, buffer);
viClose(rm);
}
```

Attribute example

```
#include <visa.h>
#include <stdio.h>
int main(int argc, char* argv[]) {
ViSession rm, vi;
ViChar buffer[256];
viOpenDefaultRM(&rm);
viOpen(rm, “GPIB0::1::INSTR”, VI_NULL, VI_NULL, &vi);
//Get VISA Manufacturer Name
viGetAttribute(vi, VI_RSRC_MANF_NAME, (ViPAttrState) buffer);
// Set Timeout to 5 seconds
viSetAttribute(vi, VI_ATTR_TMO_VALUE, 5000);
printf(“Manufacturer: %s\n”, buffer);
viClose(rm);
}
```

Exclusive lock example

```
#include <visa.h>
#include <stdio.h>
int main(int argc, char* argv[]) {
```

```
ViSession rm, vi;
ViUInt32 retCnt;
ViChar buffer[256];
viOpenDefaultRM(&rm);
viOpen(rm, “GPIB0::1::INSTR”, VI_NULL, VI_NULL, &vi);
// Locking the read/write ensures a
// second application talking to the
// same resource works as expected.
viLock(vi, VI_EXCLUSIVE_LOCK, VI_TMO_INFINITE, VI_NULL, VI_NULL);
viWrite(vi, “*idn?”, 5, &retCnt);
viRead(vi, buffer, 256, &retCnt);
viUnlock(vi);
printf(“device: %s\n”, buffer);
viClose(rm);
}
```

Formatted I/O example

```
#include <visa.h>
#include <stdio.h>
int main(int argc, char* argv[]) {
ViSession rm, vi;
ViChar buffer[256];
viOpenDefaultRM(&rm);
viOpen(rm, “GPIB0::1::INSTR”, VI_NULL, VI_NULL, &vi);
viPrintf(vi, “header off”);
viFlush(vi, VI_WRITE_BUF);
// No locking is required when
// using viQuery
viQueryf(vi, “*idn?”, “%s”, buffer);
printf(“device: %s\n”, buffer);
viClose(rm);
}
```