Porting SICL Applications to VISA

Application Note

This application note is intended to assist you in the job of porting a C or C++ SICL program to VISA. The details of the SICL and VISA function and attributes are provided in the documentation that is supplied with the Agilent IO Libraries Suite. The following on-line documents are available by navigating to the *Start button → All Programs → Agilent IO Libraries Suite → Documentation.*

- SICL Help
- VISA Help
- SICL Users Guide
- VISA Users Guide

They provide all the information necessary for porting existing SICL applications to VISA, but this application note will make the porting task a bit easier by listing, in tabular form, the VISA functions and attributes which correspond to each SICL function. This table is not intended to completely replace the need for the on-line documentation, but rather to give you a head start by pointing you in the right direction.

Notes:

1. There is not always a one-to-one correspondence between SICL and VISA functions; you will, in many cases, have to use a combination of VISA functions and attributes to perform an equivalent SICL operation.
2. Attributes of the form `VI_ATTR_`... are standard VISA attributes. You can look up details about them in Appendix B of the VISA Users Guide.
3. Attributes of the form `VI_AGATTR_`... are Agilent specific attributes. The #define values for them are defined where they are used in the table. Note that Agilent specific attributes can only be used with Agilent VISA. They are not available in other vendors' VISA implementations.
4. In the table below, for variables used in function calls, in many cases, a prefix and suffix is used to indicate the type and size of the variable. For example:

• `sData8`	indicates a signed 8 bit value
• `uData8`	indicates an unsigned 8 bit value
• `sData16`	indicates a signed 16 bit value
• `uData16`	indicates an unsigned 16 bit value
• `sData32`	indicates a signed 32 bit value.
• `uData32`	indicates an unsigned 32 bit value
• `spArray8`	indicates a pointer to an array of signed 8 bit elements

5. Rows with `gray text` indicate SICL functions that have no corresponding VISA functions.
6. Rows with `blue text` indicate unsupported SICL or VISA functions.

Agilent Technologies

SICL function	VISA function/attributes
iabort	Not documented in SICL and no corresponding VISA function

ibblockcopy	viMoveIn8(vi,memSpace,offset,length,buf8); or
	viMoveOut8(vi,memSpace,offset,length,buf8);
	-- or --
	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT,1); // this is the default
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,1); // this is the default
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_8,destSpace,destOffset,VI_WIDTH_8,length)

ibeswap	No corresponding VISA function

iblockmovex	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT, incr); // incr = 0 for fifo, 1 for block
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,incr); // incr = 0 for fifo, 1 for block
	viMove(vi,srcSpace,srcOffset,width,destSpace,destOffset,width,length);
	// width = VI_WIDTH_8, VI_WIDTH_16 or VI_WIDTH16

ibpeek	viPeek8(vi,addr,&uData8); or viIn8(vi,space,offset,&uData8);
	//The viPeek8() functions require a viMapAddress() call to set up a map.
	//The viIn8() functions do not require map. (It is done implicitly.)

ibpoke	viPoke8(vi,addr,uData8); or viOut8(vi,space,offset,uData8);
	//The viPoke8() functions require a viMapAddress() call to set up a map.
	//The viOut8() functions do not require map. (It is done implicitly.)

ibpopfifo	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT, 0);
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,1); // this is the default
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_8,destSpace,destOffset,VI_WIDTH_8,length);

ibpushfifo	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT, 1); // this is the default
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,0);
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_8,destSpace,destOffset,VI_WIDTH_8,length);

icauseerr	No corresponding VISA function
iclear	viClear(vi); for INSTR sessions viGpibSendIFC(vi); for INTFC sessions
iclose	viClose(vi);
icmd	Not documented in SICL and no corresponding VISA function
iderefptr	viGetAttribute(vi,VI_ATTR_WIN_ACCESS,&uData16);
idrvrversion	Not documented in SICL. Use: #define VI_AGATTR_DRVR_SPEC_VERSION (0x0FFF0025L) #define VI_AGATTR_DRVR_IMPL_VERSION (0x0FFF0026L) viGetAttribute(vi,VI_AGATTR_DRVR_SPEC_VERSION, &uData32); or viGetAttribute(vi,VI_AGATTR_DRVR_IMPL_VERSION. &uData32);
iflush	viFlush(vi,mask);
ifread	viBufRead(vi,buf,count,&retCount);

ifwrite	viSetAttribute(vi,VI_ATTR_SEND_END_EN, val); // set val to VI_TRUE or VI_FALSE
	viBufWrite(vi,buf,count,&retCount);

igetaddr	viGetAttribute(vi,VI_ATTR_RSRC_NAME,spArray8);
igetdata	viGetAttribute(vi,VI_ATTR_USR_DATA,&uData32);

igetdevaddr	viGetAttribute(vi,VI_ATTR_GPIB_PRIMARY_ADDR,&uData16);
	viGetAttribute(vi,VI_ATTR_GPIB_SECONDARY_ADDR,&uData16);
	viGetAttribute(vi,VI_ATTR_VXI_LA,&uData16);

igeterrno	No corresponding VISA function
igeterrstr	viStatusDesc(vi,status,spArray8); // 256 byte buffer must be allocated by caller.

igetgatewaytype	viGetAttribute(vi,VI_AGATTR_INTERFACE_PROTOCOL,&uData32);
	// To define this attribute add a '#define AGVISA_EXTENSIONS' before
	// the '#include visa.h' statement in your source file.

igetintfsess	No corresponding VISA function
igetintftype	viGetAttribute(vi,VI_ATTR_INTF_TYPE,&uData16);

igetlockwait	viSetAttribute(vi,VI_AGATTR_LOCKWAIT,&uData16);
	// To define this attribute add a '#define AGVISA_ATTRIBUTES' before
	// the '#include visa.h' statement in your source file.

SICL function	VISA function/attributes
igetlu	No corresponding VISA function
igetluinfo	No corresponding VISA function
igetlulist	No corresponding VISA function
igetonerror	No corresponding VISA function
igetonintr	No corresponding VISA function
igetonsrq	No corresponding VISA function
igetsesstype	viGetAttribute(vi,VI_ATTR_RSRC_CLASS,spArray8);

igettermchr	viGetAttribute(vi,VI_ATTR_TERMCHAR, &uData8); //(termchar value)
	viGetAttribute(vi,VI_ATTR_TERMCHR_EN,&uData16); //(enabled state)

igettimeout	viGetAttribute(vi,VI_ATTR_TMO_VALUE,&uData32);
igpibatnctl	viGpibControlATN(vi,mode); // Use VI_GPIB_ATN_ASSERT or VI_GPIB_ATN_DEASSERT for 'mode'.
igpibbusaddr	viSetAttribute(vi,VI_ATTR_GPIB_PRIMARY_ADDR,uData16); // valid on INTFC session
igpibbusstatus	Use viGetAttribute(vi,attribute,&value); to get the desired VISA attribute: **SICL request** — **VISA attribute** — **dataType** I_GPIB_BUS_REM — (no corresponding VISA attribute) I_GPIB_BUS_SRQ — VI_ATTR_GPIB_SRQ_STATE — sData16 I_GPIB_BUS_NDAC — VI_ATTR_GPIB_NDAC_STATE — sData16 I_GPIB_BUS_SYSCTLR — VI_ATTR_GPIB_SYS_CNTRL_STATE — uData16 I_GPIB_BUS_ACTCTLR — VI_ATTR_GPIB_CIC_STATE — uData16 I_GPIB_BUS_TALKER — VI_ATTR_GPIB_ADDR_STATE — sData16 I_GPIB_BUS_LISTENER — VI_ATTR_GPIB_ADDR_STATE — sData16 I_GPIB_BUS_ADDR — VI_ATTR_GPIB_PRIMARY_ADDR — uData16 I_GPIB_BUS_LINES — VI_ATTR_GPIB_REN_STATE — sData16 (not available from SICL) — VI_ATTR_GPIB_ATN_STATE — sData16* *VI_ATTR_GPIB_ATN_STATE will always be returned as VI_STATE_UNKNOWN.
igpibgett1delay	No corresponding VISA function
igpibllo	viGpibControlREN(vi,VI_GPIB_REN_ASSERT_ADDRESS_LLO); or viGpibControlREN(vi,VI_GPIB_REN_ASSERT_LLO);
igpibpassctl	viGpibPassControl(vi,primAddr,secAddr);
igpibppoll	No corresponding VISA function
igpibppollconfig	No corresponding VISA function
igpibppollresp	No corresponding VISA function
igpibpulseifc	Not documented in SICL. viGpibSendIFC uses iclear on a SICL GPIB interface session to do the equivalent.
igpibrenctl	viGpibControlREN(vi,mode); // mode: VI_GPIB_REN_ASSERT or VI_GPIB_REN_DEASSERT
igpibsendcmd	viGpibCommand(vi,buf,count,&retCount);
igpibsett1delay	No corresponding VISA function
ihint	Not documented in SICL. Used by VI_ATTR_DMA_ALLOW_EN

iintroff	No Corresponding VISA function with global action. Use:
	viEnableEvent(vi,VI_ALL_ENABLED_EVENTS,VI_SUSPEND_HANDLER,VI_NULL);
	for each VISA session.

iintron	No Corresponding VISA function with global action. Use:
	viEnableEvent(vi,VI_ALL_ENABLED_EVENTS,VI_HANDLER,VI_NULL);
	for each VISA session.

ilangettimeout	No corresponding VISA function
ilantimeout	No corresponding VISA function

ilblockcopy	viMoveIn32(vi,memSpace,offset,length,buf32); or
	viMoveOut32(vi,memSpace,offset,length,buf32);
	-- or --
	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT, 1); // this is the default
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,1); // this is the default
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_32,destSpace,destOffset,VI_WIDTH_32,length);

SICL function	VISA function/attributes
ileswap	No corresponding VISA function
ilocal	viGpibControlREN(vi,VI_GPIB_REN_ADDRESS_GTL);
ilock	viLock(vi,VI_EXCLUSIVE_LOCK,timeout,VI_NULL,VI_NULL);

ilpeek	viPeek32(vi,addr,&uData32); or viIn32(vi,space,offset,&uData32);
	//The viPeek32() functions require a viMapAddress() call to set up a map.
	//The viIn32() functions do not require map. (It is done implicitly.)

ilpoke	viPoke32(vi,addr,uData32); or viOut32(vi,space,offset,uData32);
	//The viPoke32() functions require a viMapAddress() call to set up a map.
	//The viOut32() functions do not require map. (It is done implicitly.)

ilpopfifo	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT, 0);
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,1); // this is the default
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_32,destSpace,destOffset,VI_WIDTH_32,length);

ilpushfifo	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT, 1); // this is the default
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,0);
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_32,destSpace,destOffset,VI_WIDTH_32,length);

imap	viMapAddress(vi,mapSpace,mapBase,mapSize,VI_FALSE,suggested,&address);
	// VISA allows only one map per session. If you need multiple maps, you can
	// open multiple sessions to the same device and map each session differently.

imapx	viMapAddress(vi,mapSpace,mapBase,mapSize,VI_FALSE,suggested,&address);
imapinfo	No Corresponding VISA function

ionerror	viInstallHandler(vi,VI_EVENT_EXCEPTION,handler,userHandle);
	viEnableEvent(vi,VI_EVENT_EXCEPTION,VI_HNDLR,VI_NULL);
	Use viDisableEvent(vi,VI_EVENT_EXCEPTION,VI_HNDLR); to disable.

ionintr	viInstallHandler(vi,VI_EVENT_??,handler,userHandle);
	viEnableEvent(vi,VI_EVENT_??,VI_HNDLR,VI_NULL);
	Use viDisableEvent(vi,VI_EVENT_??,VI_HNDLR); to disable.

ionsrq	viInstallHandler(vi,VI_EVENT_SERVICE_REQ,handler,userHandle);
	viEnableEvent(vi,VI_EVENT_SERVICE_REQ,VI_HNDLR,VI_NULL);
	use viDisableEvent(vi,VI_EVENT_SERVICE_REQ,VI_HNDLR); to disable

iopen	ViSession drm,vi;
	viOpenDefaultRM(&drm); // must be done once to initialize VISA before any viOpen
	viOpen(drm,openString,VI_NULL,VI_NULL,&vi);

ipeekx8	viPeek8(vi,addr,&uData8); or viIn8(vi,space,offset,&uData8);
	//The viPeek8() functions require a viMapAddress() call to set up a map.
	//The viIn8() functions do not require map. (It is done implicitly.)

ipeekx16	viPeek16(vi,addr,&uData16); or viIn16(vi,space,offset,&uData16);
	//The viPeek16() functions require a viMapAddress() call to set up a map.
	//The viIn16() functions do not require map. (It is done implicitly.)

ipeekx32	viPeek32(vi,addr,&uData32); or viIn32(vi,space,offset,&uData32);
	//The viPeek32() functions require a viMapAddress() call to set up a map.
	//The viIn32() functions do not require map. (It is done implicitly.)

ipokex8	viPoke8(vi,addr,uData8); or viOut8(vi,space,offset,uData8);
	//The viPoke8() functions require a viMapAddress() call to set up a map.
	//The viOut8() functions do not require map. (It is done implicitly.)

ipokex16	viPoke16(vi,addr,uData16); or viOut16(vi,space,offset,uData16);
	//The viPoke16() functions require a viMapAddress() call to set up a map.
	//The viOut16() functions do not require map. (It is done implicitly.)

ipokex32	viPoke32(vi,addr,uData32); or viOut32(vi,space,offset,uData32);
	//The viPoke32() functions require a viMapAddress() call to set up a map.
	//The viOut32() functions do not require map. (It is done implicitly.)

iprintf	viPrintf(vi,writeFmt,arg1,arg2,…);
ipromptf	viQueryf(vi,writeFmt,readFmt,arg1,arg2,…);
iread	viRead(vi,buf,uCount32,&uRetCount32);

SICL function	VISA function/attributes		
ireadstb	viReadSTB(vi,&uStatus16);		
iremote	viGpibControlREN(vi,VI_GPIB_REN_ASSERT_ADDRESS);		
iscanf	viScanf(vi,readFmt,arg1,arg2,…);		
iserialbreak	viClear(vi);		
iserialctrl	Use viSetAttribute(vi,attribute,value) to set desired VISA attribute:		
	SICL request	**VISA attribute**	**DataType**
	I_SERIAL_BAUD	VI_ATTR_ASRL_BAUD	uData32
	I_SERIAL_PARITY	VI_ATTR_ASRL_PARITY	uData16
	I_SERIAL_STOP	VI_ATTR_ASRL_STOP_BITS	uData16
	I_SERIAL_WIDTH	VI_ATTR_ASRL_DATA_BITS	uData16
	I_SERIAL_FLOW_CTRL	VI_ATTR_ASRL_FLOW_CNTRL	uData16
	I_SERIAL_READ_EOI	VI_ATTR_ASRL_END_IN	uData16
	I_SERIAL_WRITE_EOI	VI_ATTR_ASRL_END_OUT	uData16
	I_SERIAL_DUPLEX	(no corresponding VISA attribute)	
	I_SERIAL_WRITE_BUFSZ	(no corresponding VISA attribute)	
	SICL request	**VISA function call**	
	I_SERIAL_READ_BUFSZ	viSetBuf(vi,VI_IO_IN_BUF,size);	
	I_SERIAL_RESET	viFlush(vi,VI_IO_IN_BUF_DISCARD | VI_IO_OUT_BUF_DISCARD);	
iserialmclctrl	Use viSetAttribute(vi,attribute,value) to set desired VISA attribute:		
	SICL request	**VISA attribute**	**DataType**
	I_SERIAL_RTS	VI_ATTR_ASRL_RTS_STATE	uData16
	I_SERIAL_DTR	VI_ATTR_ASRL_DTR_STATE	uData16
iserialmclstat	Use viGetAttribute(vi,attribute,&value) to get desired VISA attribute:		
	SICL request	**VISA attribute**	**DataType**
	I_SERIAL_RTS	VI_ATTR_ASRL_RTS_STATE	uData16
	I_SERIAL_DTR	VI_ATTR_ASRL_DTR_STATE	uData16
iserialstat	Use viGetAttribute(vi,attribute,&value) to get the desired VISA attribute:		
	SICL request	**VISA attribute**	**DataType**
	I_SERIAL_BAUD	VI_ATTR_ASRL_BAUD	uData32
	I_SERIAL_PARITY	VI_ATTR_ASRL_PARITY	uData16
	I_SERIAL_STOP	VI_ATTR_ASRL_STOP_BITS	uData16
	I_SERIAL_WIDTH	VI_ATTR_ASRL_DATA_BITS	uData16
	I_SERIAL_FLOW_CTRL	VI_ATTR_ASRL_FLOW_CNTRL	uData16
	I_SERIAL_READ_EOI	VI_ATTR_ASRL_END_IN	uData16
	I_SERIAL_WRITE_EOI	VI_ATTR_ASRL_END_OUT	uData16
	I_SERIAL_DUPLEX	(no corresponding VISA attribute)	
	I_SERIAL_READ_BUFSZ	(no corresponding VISA attribute)	
	I_SERIAL_WRITE_BUFSZ	(no corresponding VISA attribute)	
	These SICL request codes return bitmaps which contain bits corresponding to various VISA attributes:		
	I_SERIAL_MSL:		
	I_SERIAL_DCD	VI_ATTR_ASRL_DCD_STATE	uData16
	I_SERIAL_DSR	VI_ATTR_ASRL_DSR_STATE	uData16
	I_SERIAL_CTS	VI_ATTR_ASRL_CST_STATE	uData16
	I_SERIAL_RI	VI_ATTR_ASRL_RI_STATE	uData16
	I_SERIAL_TERI	(no corresponding VISA attribute)	
	I_SERIAL_D_DCD	(no corresponding VISA attribute)	
	I_SERIAL_D_DSR	(no corresponding VISA attribute)	
	I_SERIAL_D_CTS	(no corresponding VISA attribute)	

SICL function	VISA function/attributes		
	I_SERIAL_STAT:		
	I_SERIAL_DAV	(no corresponding VISA attribute)	
	I_SERIAL_PARERR	(no corresponding VISA attribute)	
	I_SERIAL_OVERFLOW	(no corresponding VISA attribute)	
	I_SERIAL_FRAMING	(no corresponding VISA attribute)	
	I_SERIAL_BREAK	(no corresponding VISA attribute)	
	I_SERIAL_TEMT	(no corresponding VISA attribute)	
isetbuf	viSetBuf(vi,mask,uSize32);		
isetdata	viSetAttribute(vi,VI_ATTR_USR_DATA,uData32);		
isetintr	viEnableEvent(vi,VI_EVENT_??,VI_HNDLR,VI_NULL); // use with event handler		
	viEnableEvent(vi,VI_EVENT_??,VI_QUEUE,VI_NULL); // use with viWaitOnEvent()		
isetlockwait	viSetAttribute(vi,VI_AGATTR_LOCKWAIT,uData16);		
	// To define this attribute add a '#define AGVISA_ATTRIBUTES' before		
	// the '#include visa.h' statement in your source file.		
isetstb	viSetAttribute(vi,VI_ATTR_DEV_STATUS_BYTE,uData8);		
isetubuf	No corresponding VISA function		
isscanf	ivSScanf(vi,buf,readFmt,arg1,arg2,…);		
isprintf	viSPrintf(vi,buf,writeFmt,arg1,arg2,…);		
isvprintf	viVSPrintf(vi,buf,writeFmt,vaList);		
isvscanf	ivVSScanf(vi,buf,readFmt,vaList);		
iswap	No corresponding VISA function		
itermchr	viSetAttribute(vi,VI_ATTR_TERM_CHAR,char);		
	viSetAttribute(vi,VI_ATTR_TERM_CHAR_EN,VI_TRUE);		
itimeout	viSetAttribute(vi,VI_ATTR_TMO_VALUE,value);		
itrigger	viSetAttribute(vi,VI_ATTR_TRIG_ID,VI_TRIG_SW);		
	viAssertTrigger(vi,VI_ATTR_TRIG_PROT_DEFAULT);		
iunlock	viUnlock(vi);		
iunmap	viUnmapAddress(vi);		
iunmapx	viUnmapAddress(vi);		
iversion	viGetAttribute(vi,VI_ATTR_RSRC_IMPL_VERSION,&uData32)		
	viGetAttribute(vi,VI_ATTR_RSRC_SPEC_VERSION,&uData32)		
ivprintf	viVPrintf(vi,writeFmt,vaList);		
ivpromptf	viVQueryf(vi,buf,writeFmt,readFmt,vaList);		
ivscanf	ivVScanf(vi,readFmt,vaList);		
ivxibusstatus	Use viGetAttribute(vi,attribute,&value) to get the desired VISA attribute:		
	SICL request	**VISA attribute**	**dataType**
	I_VXI_BUS_TRIGGER	VI_ATTR_VXI_TRIG_STATUS	uData32
	I_VXI_BUS_TRIGSUPP	VI_ATTR_VXI_TRIG_SUPPORT	uData32
	I_VXI_BUS_LADDR	(no corresponding VISA attribute)	
	I_VXI_BUS_SERVANT_AREA	(no corresponding VISA attribute)	
	I_VXI_BUS_NORMOMP	(no corresponding VISA attribute)	
	I_VXI_BUS_CMDR_LADDR	(no corresponding VISA attribute)	
	I_VXI_BUS_MAN_ID	(no corresponding VISA attribute)	
	I_VXI_BUS_MODEL_ID	(no corresponding VISA attribute)	
	I_VXI_BUS_PROTOCOL	(no corresponding VISA attribute)	
	I_VXI_BUS_XPROT	(no corresponding VISA attribute)	
	I_VXI_BUS_SHM_SIZE	(no corresponding VISA attribute)	
	I_VXI_BUS_SHM_ADDR_SPACE	(no corresponding VISA attribute)	
	I_VXI_BUS_SHM_PAGE	(no corresponding VISA attribute)	
	I_VXI_BUS_VXIMXI	(no corresponding VISA attribute)	
ivxigettrigroute	No corresponding VISA function		

SICL function	VISA function/attributes
ivxirminfo	The information returned in the vxiinfo structure by the SICL ivxirminfo call is available in various VISA attributes. Use viGetAttribute(vi,attribute,&value) to get the desired VISA attribute:

vxiinfo struct member	VISA attribute	dataType
laddr	VI_ATTR_VXI_LA	uData16
manuf_name	VI_ATTR_MANF_NAME	spArray8
model_name	VI_ATTR_MODEL_NAME	spArray8
man_id	VI_ATTR_MANF_ID	uData16
model	VI_ATTR_MODEL_CODE	uData16
devclass	VI_ATTR_VXI_DEV_CLASS	uData16
cage_num	VI_ATTR_MAINFRAME_LA	data16
slot	VI_ATTR_SLOT	data16
addrspace*	VI_ATTR_MEM_SPACE	uData16
memsize*	VI_ATTR_MEM_SIZE	uData32
memstart*	VI_ATTR_MEM_START	uData32
cmdr_laddr	VI_ATTR_CMDR_LA	uData16
name	(no corresponding VISA attribute)	
selftest	(no corresponding VISA attribute)	
protocol	(no corresponding VISA attribute)	
x_protocol	(no corresponding VISA attribute)	
servant_area	(no corresponding VISA attribute)	
slot0_laddr	(no corresponding VISA attribute)	
int_handler	(no corresponding VISA attribute)	
interrupter	(no corresponding VISA attribute)	

* Memory size and start in SICL are in 'pages'. In VISA, they are in bytes.

ivxiservants	VISA does not return a list of VXI servants but: viGetAttribute(vi,VI_ATTR_IMMEDIATE_SERV,&uData16); returns whether or not the device opened on the VISA session 'vi' is an immediate servant of the controller running VISA.

ivxitrigoff	viSetAttribute(vi,VI_ATTR_TRIG_ID,uData16);
	viAssertTrigger(vi,VI_TRIG_PROT_OFF);

ivxitrigon	viSetAttribute(vi,VI_ATTR_TRIG_ID,uData16);
	viAssertTrigger(vi,VI_TRIG_PROT_ON);

ivxitrigroute	viMapTrigger(vi,trigSrc,trigDest,VI_NULL); // sets up a trigger mapping
	viUnmapTrigger(vi,trigSrc,trigDest); // unmaps the src and dest triggers.

ivxiwaitnormop	No corresponding VISA function
ivxiws	viVxiCommandQuery(vi,mode,cmd,&response);

iwaithdlr	viEnableEvent(vi,VI_EVENT_??,VI_QUEUE,VI_NULL); // enables selected event type
	viWaitOnEvent(vi,VI_EVENT_??,uTimeout32,&eventType,&outContext);
	// if viWaitOnEvent did not timeout it returns a context in the 'outContext'
	// (unless you specified VI_NULL for that parameter. You can query attributes
	// of the context using viGetAttribute(outContext,attrib,&value);.
	// When finished with the context, do viClose(outContext);.

iwblockcopy	viMoveIn16(vi,memSpace,offset,length,buf16); or
	viMoveOut16(vi,memSpace,offset,length,buf16);
	-- or --
	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT,1); // this is the default
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,1); // this is the default
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_16,destSpace,destOffset,VI_WIDTH_16,length);

iwpeek	viPeek16(vi,addr,&uData16); or viIn16(vi,space,offset,&uData16);
	//The viPeek16() functions require a viMapAddress() call to set up a map.
	//The viIn16() functions do not require map. (It is done implicitly.)

SICL function	VISA function/attributes

iwpoke	viPoke16(vi,addr,uData16); or viOut16(vi,space,offset,uData16);
	//The viPoke16() functions require a viMapAddress() call to set up a map.
	//The viOut16() functions do not require map. (It is done implicitly.)

iwpopfifo	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT,0);
	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,1); // this is the default
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_16,destSpace,destOffset,VI_WIDTH_16,length);

iwpushfifo	viSetAttribute(vi,VI_ATTR_SRC_INCREMENT,1); // this is the default

	viSetAttribute(vi,VI_ATTR_DEST_INCREMENT,0);
	viMove(vi,srcSpace,srcOffset,VI_WIDTH_16,destSpace,destOffset,VI_WIDTH_16,length);

iwrite	viSetAttribute(vi,VI_ATTR_SEND_END_EN, ??); // set to VI_TRUE or VI_FALSE
	viWrite(vi,buf,count,&retCount);

ixtrig	viSetAttribute(vi,VI_ATTR_TRIG_ID,VI_TRIG_??);
	viAssertTrigger(vi,VI_ATTR_TRIG_PROT_DEFAULT);

_siclcleanup	This functionality is not required in VISA. (In SICL it was only needed for 16-bit code, which is no longer supported.)

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

Agilent Open

www.agilent.com/find/open
Agilent Open simplifies the process of connecting and programming test systems to help engineers design, validate and manufacture electronic products. Agilent offers open connectivity for a broad range of system-ready instruments, open industry software, PC-standard I/O and global support, which are combined to more easily integrate test system development.

LXI

www.lxistandard.org
LXI is the LAN-based successor to GPIB, providing faster, more efficient connectivity. Agilent is a founding member of the LXI consortium.

Agilent T&M Software and Connectivity
Agilent's Test and Measurement software and connectivity products, solutions and Agilent Developer Network allow you to take time out of connecting your instrument to your computer with tools based on PC standards, so you can focus on your tasks, not on your connections. Visit:
www.agilent.com/find/connectivity.

Remove all doubt

Our repair and calibration services will get your equipment back to you, performing like new, when promised. You will get full value out of your Agilent equipment throughout its lifetime. Your equipment will be serviced by Agilent-trained technicians using the latest factory calibration procedures, automated repair diagnostics and genuine parts. You will always have the utmost confidence in your measurements.

Agilent offers a wide range of additional expert test and measurement services for your equipment, including initial start-up assistance onsite education and training, as well as design, system integration, and project management.

For more information on repair and calibration services, go to

www.agilent.com/find/removealldoubt

www.agilent.com

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:
www.agilent.com/find/contactus

Phone or Fax

Americas	
Canada	877 894 4414
Latin America	305 269 7500
United States	800 829 4444

Asia Pacific	
Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	81 426 56 7832
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe	
Austria	0820 87 44 11
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700
Germany	01805 24 6333*
	*0.14€/minute
Ireland	1890 924 204
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland (French)	44 (21) 8113811 (Opt 2)
Switzerland (German)	0800 80 53 53 (Opt 1)
United Kingdom	44 (0) 7004 666666

Other European Countries:
www.agilent.com/find/contactus

Revised: March 23, 2007

Microsoft is a U.S. registered trademark of Microsoft Corporation.

Product specifications and descriptions in this document subject to change without notice.

© Agilent Technologies, Inc. 2007
Printed in USA, March 30, 2007
5989-6581EN

Agilent Technologies