Agilent I/O Libraries Technical Overview

Agilent Technologies

The Agilent I/O Libraries are the software that is common to the Agilent I/O hardware products -- the original Standard Instrument Control Library (SICL), and the public-standard Virtual Instrument Standard Architecture (VISA).

SICL is an Agilent standard I/O library which preceded the VXI*plug&play* standard VISA. It is provided to support exiting SICL applications. While SICL offers the steps forward of being supported on multiple platforms and multiple interfaces, it is an HP-only standard. The next step forward is a standard supported by multiple vendors. This was accomplished through the VISA spec, defined by the VXI *Plug&Play* Alliance. The following describes these I/O Libraries.

VISA

This is an I/O Library similar in features and functionality to SICL. It was developed by the VXIp*lug&play* System alliance to specify a standard that can be used with multiple vendors. Like SICL, it supports multiple platforms, multiple interfaces -- and has similar capabilities, but different syntax. Agilent VISA is implemented on top of SICL. Note that Agilent released an early version of VISA on Win3 that was known as the VISA Transition Library, or VTL. VTL is obsolete.

VISA is required for support of VXIplug&play drivers and is currently also supported under Microsoft and Borland C/C++. It also works on Visual BASIC for Windows. VISA resembles SICL in many ways but with many differences. The best way to get started with VISA is to take a look at an example program and then discuss its details:

```
#define DMM "GPIB0::22::INSTR"       /* Instrument address (722). */
#define TIMEOUT 5000                 /* Timeout in milliseconds. */

#include <visa.h>
#include <stdio.h>
#include <stdlib.h>

void main ()
{

  ViSession rsm,
            dev;
  ViStatus  err;
  char      b[256];

  puts( "Opening session and setting timeout." );

  err = viOpenDefaultRM( &rsm );
  if( err != 0 )
  {
    puts( "Can't open resource manager." );
    exit( 1 );
  }
  err = viOpen( rsm, DMM, VI_NULL, VI_NULL, &dev );
  if( err != 0 )
  {
    puts( "Can't open device session." );
    exit( 1 );
```

```
  }
  viSetAttribute( dev, VI_ATTR_TMO_VALUE, TIMEOUT );

  puts( "Initializing device and getting ID string from DMM." );

  err = viClear( dev );
  if( err != 0 )
  {
    puts( "Can't clear interface." );
    exit( 1 );
  }
  err = viPrintf( dev, "*RST;*CLS\n" );
  if( err != 0 )
  {
    puts( "Can't send reset-clear commands to device." );
    exit( 1 );
  }
  err = viPrintf( dev, "*IDN?\n" );
  if( err != 0 )
  {
    puts( "Can't send ID command to device." );
    exit( 1 );
  }
  err = viScanf( dev, "%s", &b );
  if( err != 0 )
  {
    puts( "Can't read ID string from device." );
    exit( 1 );
  }

  printf( "ID:  %s\n", b );

  puts( "Closing session, all done." );

  viClose( dev );
  viClose( rsm );
}
```

The first thing to notice is that to compile a VISA-based program you need to "include" the appropriate header file:

```
include "visa.h"
```

In SICL, you have to open a session on a particular interface; this is true for VISA as well, but you also have open a session for the "VISA resource manager" -- a software process analogous to (but different in detail from) a VXI resource manager. A particular device is identified with yet another new device syntax -- in this case:

```
GPIB0::22:INSTR
```

-- gives the device at address 22 on the default interface. This addressing scheme follows the form:

```
GPIB0::8::INSTR           708
GPIB0::8::4::INSTR        70804
ASLR1::INSTR              COM1
VXI0::24::INSTR           Backplane access to logical address 24.
GPIB-VXI0::24::INSTR      Command module at 70900 & device at 24.
```

Note that:

- GPIBn indicates a GPIB interface to an instrument using GPIB primary/secondary addresses.
- ASLR1 indicates a serial port.
- VXIn indicates embedded-controller (or similar) access to a VXI mainframe using VXI logical addresses.
- GPIB-VXIn indicates access to a VXI mainframe through a GPIB command module ... this might seem to be equivalent to the "GPIBn" designation, but the "GPIB-VXIn" scheme allows you to still use VXI logical addresses and VXI-backplane specific operations through an emulation scheme.

A VISA program looks much like a SICL program, though you can't install an errorhandler -- you have to check for errors yourself. (The error variable will be less than 0 for errors, will equal zero if no error, and will be greater than 0 for warnings.) You also have to use a single set of commands, "viSetAttribute()" and "viGetAttribute()", to perform various control and status functions; for example:

```
viSetAttribute( dev, VI_ATTR_TMO_VALUE, TIMEOUT );   // Set timeout.
viSetAttribute( dev, VI_ATTR_SEND_END_EN, VI_TRUE );   // Enable END.
viSetAttribute( dev, VI_ATTR_SUPPRESS_END_EN, TIMEOUT );   // Set timeout.
viSetAttribute( dev, VI_ATTR_TERMCHR, '\n' );   // Set termchar.
viSetAttribute( dev, VI_ATTR_TERMCHR_EN, VI_FALSE );  // Disable termchar.
```

Other features of VISA not shown here include an ability to read the currently known devices from the VISA resource manager, and the ability to either wait for or set up a handler for events. VISA implements the following commands:

```
viAssertTrigger   // Assert interface trigger.
viClear   // Clear a device.
viClose   // Close a session.
viDisableEvent   // Disable event handling.
viDiscardEvents   // Discard logged events.
viEnableEvent   // Enable event.
viEventHandler   // Define event handler.
viFindNext   // Get next device from list created by "viFindRsrc()".
viFindRsrc   // Get list of devices on a specified interface.
viFlush   // Flush read and write buffers associated with formatted I/O.
viGetAttribute   // Get session attribute value.
viIn8 and viIn16   // Get byte/word value from memory space.
viInstallHandler   // Install handler routine for event callbacks.
viMapAddress   // Map memory address space.
viOpen   // Open a session to specific device.
viOpenDefaultRM   // Open a session to default resource manager.
viOut8 and viOut16   // Write byte/word to memory address.
viPeek8 and viPeek16   // Read byte/word from memory address.
viPoke8 and viPoke16   // Write byte/word to memory address.
viPrintf   // Perform formatted output.
viRead   // Perform unformatted input.
viReadSTB   // Read status byte from device that requests service.
viScanf   // Perform formatted input.
viSetAttribute   // Set session attribute.
viSetBuf   // Set buffer size for formatted I/O.
viStatusDesc   // Provide status string.
viUninstallHandler   // Remove event handler.
viUnmapAddress   // Unmap memory address.
viVPrintf   // Perform formatted output.
viVScanf   // Perform formatted input.
viWaitOnEvent   // Wait for event to occur.
viWrite   // Write unformatted data to device.
```

Some final comments:

- The 82335 card cannot be used with VISA (which implies that it cannot be used with VXIplug&play drivers).
- Beware of mixing VISA and SICL calls in the same program; you cannot communicate with the same instrument using both VISA and SICL at the same time, you must close all SICL sessions before closing the last VISA session, and you must not try to access VISA sessions with SICL calls, or the reverse.
- With VISA 2.0 or less, the LOCAL/REMOTE or PASS CONTROL features are not implemented.

SICL

The SICL software provides a library of I/O routines that has considerable flexibility and is source-compatible (to the extent possible) between different platforms and different interfaces. SICL is supported under C (as well as VEE and BASIC for HP-UX) on HP-UX platforms, and C and Visual BASIC (as well as VEE and BASIC for Windows) on Microsoft Windows platforms. It supports GPIB, RS-232, LAN-GPIB, and GPIO interfaces. Since SICL is an Agilent only library, it is only recommend for existing SICL applications. VISA is recommend for new applications.

A C program using SICL to get a voltage measurement from a DMM is illustrated below:

```
#include <stdio.h>
#include "sicl.h"

#define VOLTMETER "hpib7,22"

void main(void)
{
   INST dvm;
   float data;

   ionerror(I_ERROR_EXIT);
   dvm = iopen(VOLTMETER);
   itimeout(dvm,1000);

   ipromptf(dvm,"MEAS:VOLT:DC?\n", "%f", &data);

   printf("Result is %f\n", data);
   iclose(dvm);

}
```

The SICL commands are listed below. Note that some commands are specific to certain interfaces, and some are not implemented for Visual BASIC:

```
IABORT                        Abort current SICL operations.
IBLOCKCOPY                    Perform block data copy.
ICAUSEERR                     Force SICL error.
ICLEAR                        Clear device or interface.
ICLOSE                        Close SICL session.
IFLUSH                        Flush formatted-I/O buffers.
IFREAD                        Read formatted-I/O data.
IFWRITE                       Write formatted-I/O data.
IGETADDR                      Returns address of string sent to iopen.
IGETDATA                      Get pointer to data structure for isetdata.
IGETDEVADDR                   Get address of remote device.
IGETERRNO                     Get error number of last error.
IGETERRSTR                    Get error string for error number.
IGETGATEWAYTYPE  NO-VB  LAN   Get LAN gateway for session.
```

```
   IGETINTFSESS                Get interface session number.
   IGETINTFTYPE                Get interface type for session.
   IGETLOCKWAIT                Get locking wait flag setting.
   IGETLU                      Get interface address.
   IGETLUINFO                  Get interface description.
   IGETLULIST                  Get list of configured interface.
   IGETONERROR      NO-VB      Get error handler address.
   IGETONINTR       NO-VB      Get interrupt handler address.
   IGETONSRQ        NO-VB GPIB Get SRQ handler address.
   IGETSESSTYPE                Get session type (interface, dev, commander).
   IGETTERMCHR                 Get termination character.
   IGETTIMEOUT                 Get current timeout value.
   IGPIBATNCTL           GPIB  Control ATN line.
   IGPIBBUSADDR          GPIB  Set GPIB card address.
   IGPIBBUSSTATUS        GPIB  Get GPIB bus line status.
   IGPIBGETT1DELAY       GPIB  Get T1 delay time value.
   IGPIBLLO              GPIB  Set GPIB local lockout mode.
   IGPIBPASSCTL          GPIB  Pass control.
   IGPIBPPOLL            GPIB  Perform GPIB parallel poll.
   IGPIBPPOLLCONFIG      GPIB  Set up parallel poll configuration.
   IGPIBPPOLLRESP        GPIB  Parallel poll response value.
   IGPIBRENCTL           GPIB  Control GPIB remote enable line.
   IGPIBSENDCMD          GPIB  Sent GPIB commands.
   IGPIBSETT1DELAY       GPIB  Set T1 delay time value.
   IHINT                       Specify transfer mode (DMA, POLL, INTR).
   IINTROFF         NO-VB      Disable interrupt handlers.
   IINTRON          NO-VB      Enable interrupt handlers.
   ILANGETTIMEOUT   NO-VB LAN  Get LAN timeout value.
   ILANTIMEOUT      NO-VB LAN  Set LAN timeout value.
   ILOCAL                GPIB  Set local operation.
   ILOCK                       Lock a session.
   IMAP                  VXI   Map VXI memory into process space.
   IMAPINFO              VXI   Determine available VXI memory mapping
options.
   IONERROR         NO-VB      Install error handler.
   IONINTR          NO-VB      Install interrupt handler.
   IONSRQ           NO-VB      Install SRQ handler.
   IOPEN                       Open an interface session.
   IPEEK                 VXI   Peek from VXI memory space.
   IPOKE                 VXI   Poke into VXI memory space.
   IPOPFIFO              VXI   Read data from FIFO and put into memory.
   IPRINTF                     Perform printed output.
   IPROMPTF                    Send string and get response.
   IPUSHFIFO             VXI   Read data from memory and put into FIFO
   IREAD                       Get raw data from device.
   IREADSTB                    Read SRQ status byte.
   IREMOTE               GPIB  Set remote mode.
   ISCANF                      Read formatted data from device.
   ISERIALBREAK          SER   Send break character over serial.
   ISERIALCTRL           SER   Set serial parameters.
   ISERIALMCLCTRL        SER   Set serial control lines.
   ISERIALMCLSTAT        SER   Read status of serial control lines.
   ISERIALSTAT           SER   Read status of serial parameters.
   ISETBUF          NO-VB      Set buffers for formatted I/O.
   ISETDATA         NO-VB      Set up user-defined data structures.
   ISETINTR         NO-VB      Enable response to interrupt.
   ISETLOCKWAIT                Set wait or error on locked device.
```

ISETSTB		GPIB	Send SRQ response byte.
ISETUBUF	NO-VB		Set up buffer for formatted I/O.
ISWAP			Swap data byte ordering in a block.
ITERMCHR			Set output termination character.
ITIMEOUT			Set timeout value.
ITRIGGER			Trigger remote device.
IUNLOCK			Unlock a device or interface.
IUNMAP		VXI	Unmap VXI memory mapping.
IVERSION			Get SICL revision level.
IVXIBUSSTATUS		VXI	Get VXI bus status.
IVXIGETTRIGROUTE		VXI	Get VXI trigger routing.
IVXIRMINFO		VXI	Get data on device from resource manager.
IVXISERVANTS		VXI	Get list of servants.
IVXITRIGOFF		VXI	Disable VXI triggering.
IVXITRIGON		VXI	Enable VXI triggering.
IVXITRIGROUTE		VXI	Specify VXI trigger routing.
IVXIWAITNORMOP		VXI	Specify wait or error on VXI operation.
IVXIWS		VXI	Send VXI word-serial command.
IWAITHDLR	NO-VB		Wait on handler execution.
IWRITE			Write unformatted data.
IXTRIG			Execute extended trigger.
SICLCLEANUP			Performs housekeeping (Win3 only).