ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

An Example C Language Program for sending SCPI commands to a VXI-11 Instrument

INTRODUCTION

This application bulletin describes how to communicate and control a VXI-11 instrument using SCPI commands from a C language program in a Intel-Windows PC. The task is to provide an example of sending SCPI commands to a VXI-11 instrument to control bits in its digital interface. This example utilizes a VISA library to handle the communication between the program and the VXI-11 instrument.

GENERAL CONCEPTS

The instrument chosen is an ICS' 8063 Ethernet to Digital Interface. The Model 8063 has a 48-line digital interface that can be user configured as inputs or outputs in 8-bit bytes. The 8063 is an VXI-11.3 compatible instrument that communicates over the network using RPC or the VXI-11 protocol. The VXI-11 protocol provides GPIB like functionality over an Ethernet network. For more information about the VXI-11 Specification and the VXI-11 communication protocol, see AB80-11

There are two ways for a Windows PC to communicate with a VXI-11 instrument. One is to use the industry standard ONC RPC[1] protocol. Unfortunately Microsoft's RPC does not comply with the ONC RPC used in VXI-11 instruments and Linux/Unix systems so anyone wishing to use RPC on a PC would have to install a third party's RPC package. The second alternative is to use an a VXI-11 compatible VISA library that converts program calls to the VXI-11 protocol. VXI-11 compatible VISA libraries are available from Agilent and National Instruments. This example uses the VISA library from National Instruments.

Notes: 1. ONC RPS stands for Open Network Computing Remote Procedure Calls developed by Sun Microsystems. See Reference 3.

PREPARATION

For this example, you will need a VISA runtime system installed on your PC. On a Win32 system, this is a DLL library and is normally found in the \Windows\System32 folder. The National Instruments VISA library is named VISA-32.DLL.

If you do not have a VISA library on your PC, download a copy from National Instrument's website. VISA specific functions are noted with "vi" preceding the function name. For more information on VISA commands please reference the documentation specific to the VISA you are using.

The user must supply the VISA.H header file which defines the VISA functions and constants. In addition, the user must supply the VISA OBJ (or LIB) file to be linked to the example. The OBJ (or LIB) contains the stubs required to invoke the VISA DLL library at runtime.

PROGRAM DESCRIPTION

The code is divided into two main sections, the first section deals with establishing communication and identifying your instrument. All IEEE-488.2 and VXI-11 compatible devices will respond to an *IDN query with a unique text string that identifies the device. Thus a simple communication test consists of writing the minimal VISA functions required to open a session to the device and then perform an *IDN query and then reading back the response. This confirms that bi-directional dataflow is possible and that the correct device is responding.

After confirming that a VISA communication has been established we move into the second section of code. Five consecutive SCPI commands are sent via VISA write(viWrite) command. These commands are examples of byte and bit commands setting the inputs and/or outputs. Once all the command are executed then the VISA resource will be closed and the program will wait for any key to be pressed before exiting the program. This allows you to review any error messages that may have appeared during runtime. In a test application, the VISA resource(s) should only be closed when exiting the program.

EXAMPLE CODE

Figure 1 lists an example C language program that calls the National Instruments' VISA library to communicate with a VXI-11 instrument. Using the VISA commands to communicate we can send SCPI commands to the instruments interface. A zip file with the complete project can be downloaded from the ICS Electronics website as SCPI_Example.zip.

This example code uses ICS's default IP address for Ethernet instruments. Change the IP address to match your instrument or change the instrument's IP address prior to running the program. See the instrument's manual for instructions on changing its IP address[5]

Text continues on page 3.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "visa.h"

#define WaitForKeystroke   {while (!_kbhit()) {}}

static char outputBuffer[VI_FIND_BUFLEN];
static ViSession defaultRM, instr;
static ViStatus status;
static ViUInt32 count;
static ViUInt16 portNo;


//TargetIP can be changed here to locate your own device.
//Current Example Works with ICS Electronics default settings.
static char targetIP[] = "TCPIP0::192.168.0.254::inst0::INSTR"; //

//SCPI commands used in VIWrite command.
//These can be changed to fit your needs

//Sets Port 4 Outputs to low polarity true
static char command0[] = "SOURce:DATA:PORT4:POLarity #h00";

//Sets Port 4 Outputs to zero
static char command1[] = "SOURCE:DATA:PORT4 0";

//Outputs a one on port 4 bit 1
static char command2[] = "ROUTe:CLOSe 4,1 ";

//Sets Port 1 Inputs to High polarity true
static char command3[] = "SENSE:DATA:PORT1:POLARITY #HFF";

//Reads Port 1
static char command4[] = "SENSE:DATA:PORT1?";
```

```
int main(void)
{
char result[100];

  // First we will need to open the default resource manager.
  status = viOpenDefaultRM (&defaultRM);
  if (status < VI_SUCCESS)
  {
     printf("Could not open a session to the VISA Resource Manager!\
n");
    printf("Press any key to continue\n");
    WaitForKeystroke
    return(1);
  }

  // Now we will open a session via TCP/IP
  // TargetIP can be modified via static char listed above.
  status = viOpen (defaultRM, targetIP, 0, 0, &instr);
  if (status < VI_SUCCESS)
  {
    viClose(defaultRM);
    printf ("An error occurred opening the session to %s\n",targetIP);
    printf("Press any key to continue\n");
    WaitForKeystroke
    return(1);
  }

  //Sets Delay Time
  viSetAttribute (instr, VI_ATTR_TCPIP_NODELAY, 1);



   //Uses the viWrite command to write an "*idn?" message.
   //See viWrite command for more information.
   status = viWrite (instr, "*idn?\n", 6, &count);
   if (status < VI_SUCCESS)
   {
     viClose(defaultRM);
     printf("viWrite *idn? failed with error code %x \n",status);
     printf("Press any key to continue\n");
     WaitForKeystroke
     return(1);
  }

  //Reads reply and saves reply in result.
  status = viRead (instr, result, 100, &count);
  if (status < VI_SUCCESS)
  {
    viClose(defaultRM);
    printf("viRead failed with error code %x \n",status);
    WaitForKeystroke
    return(1);
  }
   //places end of string at the end of the IDN reply
   result[count] = '\0';

  //prints reply message to "*IDN?"
  printf("The server response is:\n %s\n\n", result);

/*---------------------------------------------------------*/
```

Figure 1 C Language Example continued

```
/*          //------SCPI Commands------//          */
/*--------------------------------------------------------*/

     printf("Press any key to start SCPI commands\n");
     WaitForKeystroke

     // command0-4 are SCPI commands listed at the top.
     //Sets Port 4 Outputs to low polarity true
     status = viWrite (instr, command0, sizeof(command0), &count);
     if (status < VI_SUCCESS)
     {
       printf("viWrite command0 failed with error code %x \n",status);
       printf("Press any key to continue\n");
       WaitForKeystroke
      }

     //Sets Port 4 Outputs to zero
     status = viWrite (instr, command1, sizeof(command1), &count);
     if (status < VI_SUCCESS)                    //Error Checking
     {
       printf("viWrite command1 failed with error code %x \n",status);
       printf("Press any key to continue\n");
       WaitForKeystroke
     }


                          //Sets Port 4 Outputs to zero
     status = viWrite (instr, command1, sizeof(command1), &count);
     if (status < VI_SUCCESS)                    //Error Checking
     {
       printf("viWrite command1 failed with error code %x \n",status);
       printf("Press any key to continue\n");
       WaitForKeystroke
     }
     //Outputs a one on port 4 bit 1
     status = viWrite (instr, command2, sizeof(command2), &count);
     if (status < VI_SUCCESS)                    //Error Checking
     {
       printf("viWrite command2 failed with error code %x \n",status);
       printf("Press any key to continue \n");
       WaitForKeystroke
      }

     //Sets Port 1 Inputs to High polarity true
     status = viWrite (instr, command3, sizeof(command3), &count);
     if (status < VI_SUCCESS)                    //Error Checking
     {
       printf("viWrite command3 failed with error code %x \n",status);
       printf("Press any key to continue \n");
       WaitForKeystroke
      }

     //Reads Port 1
     status = viWrite (instr, command4, sizeof(command4), &count);
     if (status < VI_SUCCESS)                    //Error Checking
     {
       printf("viWrite command4 failed with error code %x \n",status);
       printf("Press any key to continue \n");
       WaitForKeystroke
      }

     //Closes all resources
     status = viClose (instr);
     status = viClose (defaultRM);
     printf ("\nHit any key to exit.");
     WaitForKeystroke
   return 0;
}
```

Figure 1 C Language Example continued

SUMMARY

This Application Note has described how to develop a 'C' language program that can use SCPI commands to control the digital interface on an ICS 8063 Ethernet to Digital Interface. The example program was limited to a few commands that toggled the I/O lines and read back the state of the I/O lines. The example can be expanded by adding more commands to the program. This example applies to all ICS 80x3 products. The program can be applied to other instruments by changing the SCPI commands to those that apply to the new instrument.

REFERENCES

The following references provide more information on the subjects discussed in this Application Note:

[1] IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.

[2] IEEE Std 488.2-1992, IEEE Standard Codes, Formats, Protocols, and Common Commands For Use With IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.

[3] RPC: Remote Procedure Call Protocol Specification, Request for Comments 1057, Sun Microsystems, DDN Network Information Center, SRI International, June, 1988.

[4] VXI-11 description based on VXI-11 Specification Revision 1.0 dated 1995. Copies available from the VXI Consortium at http://www.vxi.org/?q=node/206

[5] ICS 8063 SCPI commands from 8063 Instruction Manual. Copies available from http://www.icselect.com/