ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

VXI-11 TUTORIAL
and RPC Programming Guide

INTRODUCTION

Mention the VXI-11 Specification and you will get a lot of blank looks since the VXI-11 Specification is not as well known as IEEE-488 Standard or SCPI. A lot of engineers think that the VXI-11 Specification deals exclusively with VXIbus Systems which is not true. Because of the increasing use of instruments with LAN (Ethernet) interfaces, knowledge about the VXI-11 Communication Protocol is becoming more important. This Application Note describes the specification and shows the user how to use the VXI-11 protocol to control instruments.

THE VXI-11 SPECIFICATION

The VXI-11 Specification was part of a suite of specifications developed in the early 1990s when the VXIbus concept was created. VXI-11 describes how instruments or other devices can be connected to industry-standard TCP/IP networks. The communications and programming paradigms supported by the VXI-11 specification are similar in nature to the techniques supported by IEEE-488.1 and IEEE 488.2. The protocol described allows ASCII-based communications to take place between a controller and a device over a computer network.

The VXI-11 Specification has the following objectives:

1. To allow ASCII messages, including IEEE 488.2 messages, and IEEE 488.1 instrument control messages to be passed between a controller and a device over a TCP/IP network.
2. To define an instrument protocol which can be used for this controller/device communication over a TCP/IP network.
3. To enable the interconnection of independently manufactured apparatus into a single functional system.
4. To provide a mechanism to extend the protocol.
5. To define an instrument protocol which can support diverse application interfaces.
6. To allow for other networking protocols as the functionality of devices and controllers dictate, such as NFS or telnet.

The VXI-11 Specification has three sub-sections:

VXI-11.1 which deals with connecting VXIbus devices to a network and is not considered any further by this Application Note.

VXI-11.2 deals with connecting GPIB instruments to a network though a Gateway like ICS's 8065 or Agilent's E5810A Ethernet-to-GPIB Controller.

VXI-11.3 deals with connecting IEEE-488.2 instruments with Ethernet interfaces directly to a network. A couple of examples are ICS's 8064 Relay Interface or 8099 Modbus RTU Controller.

The VXI-11.2 and VXI-11.3 sub-sections are the primary focus of this Application Note.

NETWORK INSTRUMENT PROTOCOL

In order to allow ASCII messages and IEEE-488.1 instrument control messages to pass over a TCP Network, the VXI-11 Specification created the Instrument Messages listed below in Table 1. These messages are based on the ONC Remote Procedure Call (RPC) model. This model allows an application (typically called the client) to call procedures in a remote application or device (the server) as if the remote procedures were being executed locally. The Instrument Messages will be familiar to anyone who has worked with GPIB instruments. (See ICS Application Bulletin AB48-11)

Table 1 VXI-11 Instrument Messages

Message	Channel	Description
create_link	core	opens a link to a device
device_write	core	device receives a message
device_read	core	device returns a result
device_readstb	core	device returns its status byte
device_trigger	core	device executes a trigger
device_clear	core	device clears itself
device_remote	core	device disables its front panel
device_local	core	device enables its front panel
device_lock	core	device is locked
device_unlock	core	device is unlocked
create_intr_chan	core	device creates interrupt channel
destroy_intr_chan	core	device destroys interrupt channel
device_enable_srq	core	device enables/disables sending of service requests
device_docmd	core	device executes a command
destroy_link	core	closes a link to a device
device_abort	abort	device aborts an in-progress call
device_intr_srq	interrupt	used by device to send a service request

CHANNELS AND LINKS

Figure 1 shows the basic channel connection concept. A channel is a connection from an instrument controller (client) to an instrument device (server). The Core channel is the main channel for communicating with a VXI-11.2 Gateway or a VXI-11.3 Instrument. The Core channel is used for all RPCs except for aborts and interrupts

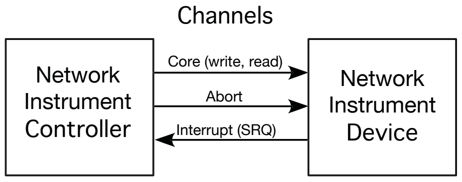

Figure 1 Network Instrument Channels

The Abort channel is a secondary control channel that is used to tell the Device to abort the current Core channel operation. The Interrupt channel is used by the Device to send Service Requests to the Application. However, in the case of the Interrupt channel, the Device acts as the client and sends the request to a server process in the Controller.

Links represent an instance of a communication pathway between a controller and a device. Any given network instrument connection (channel) may carry multiple links created with the create_link RPC. Also note that more than one controller may have a link open to a single device at the same time. If this is a possibility, then you can use a device lock to block other controllers from accessing the device and interfering with its operation.

Figure 2 shows the basic one link connection.

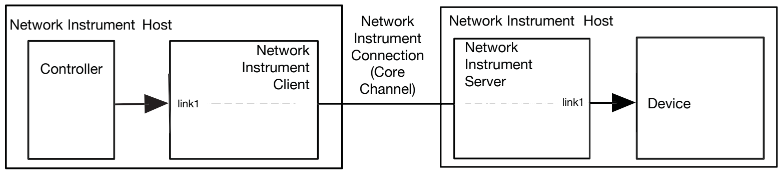
Network Instrument Host
Controller
Network Instrument Client
link1
Network Instrument Connection (Core Channel)
Network Instrument Host
Network Instrument Server
link1
Device

Figure 2 Basic Single Device Link

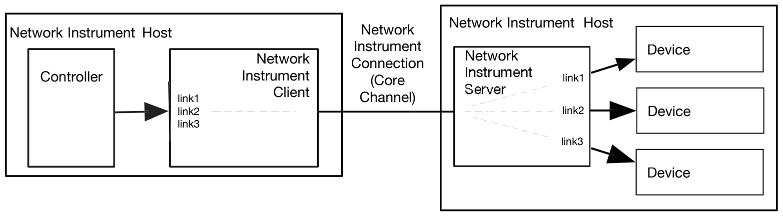

Figure 3 Multiple Link Example

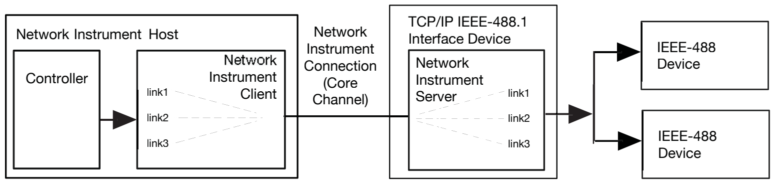
Network Instrument Host
Controller
Network Instrument Client
link1
link2
link3
Network Instrument Connection (Core Channel)
TCP/IP IEEE-488.1 Interface Device
Network Instrument Server
link1
link2
link3
IEEE-488 Device
IEEE-488 Device

Figure 4 Gateway to GPIB Instrument Link Example

Figure 3 shows an example of a controller having links to multiple logical devices in a single instrument. This could be an Ethernet instrument with three addressable sub-sections. If the single instrument was a GPIB instrument it might have a single primary address with multiple secondary addresses. Most VXI-11 devices use *inst0* as the logical device name. In Figure 3, the logical device names would be *inst0*, *inst1* and *inst2*.

Figure 4 shows an example of an instrument controller linked to two GPIB instruments through a GPIB Gateway. GPIB Gateways recognize *gpib0* as the interface name for the first IEEE-488.1 interface in the Gateway. GPIB devices attached to the Gateway are referred to by adding their GPIB address to the Gateway's interface name.

link1 can be to *gpib0* - the Gateway
link2 can be to *gpib0,4* - a GPIB device with a primary address of 4
link3 can be to *gpib0,6,11* - a GPIB device with a primary address of 6 and a secondary address of 11.

PROGRAMMING VXI-11 DEVICES

There are two major ways to program VXI-11 Devices: make calls to a VXI-11 compliant VISA library or install the VXI-11's RPCL in your computer and write your program with RPC calls. Windows users will find it easier to use a VISA library which has a VXI-11 TCP/IP output and program with graphical programs like National Instruments' LabVIEW or Agilent's VEE or make VISA calls from familiar programming languages such as C/C++ or Visual Basic. UNIX, LINUX and similar operating system users (Sun, Apple etc.) will find it easier to install the VXI-11 RPCL on the computer and then to write the application program in C/C++. VISA libraries are also available for some UNIX like operating systems but they often have driver problems and it is best to avoid them.

VXI-11 compliant VISA libraries are available from National Instruments and Agilent. National Instruments claims their VISA is VXI-11 compliant but it only supports VXI-11.3 RPCs so it can only be used with VXI-11.3 instruments. It cannot be used to control GPIB bus functions or to completely control VXI-11.2 GPIB Gateways such as Agilent's E5810A or ICS's 8065. The NI Enet/100 is not a VXI-11 Gateway. Agilent's VISA supports VXI-11.2 and 11.3 RPCs so it can be used with GPIB Gateways and VXI-11.3 instruments.

Agilent includes their VISA and SICL libraries as part of their IO Libraries Suite. SICL is the workhorse library inside Agilent's VISA. It is fully VXI-11.2 and 11.3 capable and has been stable for many years. Agilent has numerous Application Notes describing how to program with their VISA and SICL library. Also see ICS Application Bulletin AB80-2 for an interactive SICL keyboard program written in Visual Basic.

Figure 5 shows how easy it is for Windows programmers to control VXI-11 instruments through a VISA layer. Figure 5A shows how to control GPIB Instruments through a GPIB Gateway like the ICS 8065. A C/C++, Visual Basic or LabVIEW programmer only has to set the resource string to a TCPIP device to talk to the Gateway and to the instruments.

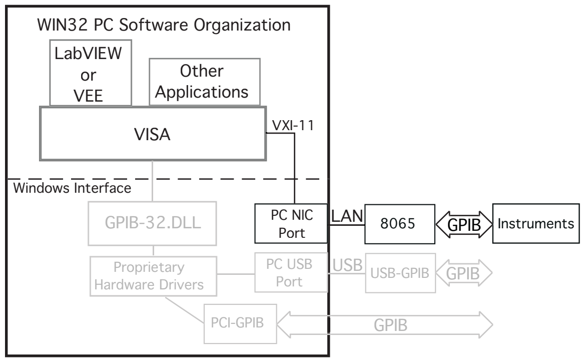

Figure 5A Controlling GPIB Devices with a GPIB Gateway

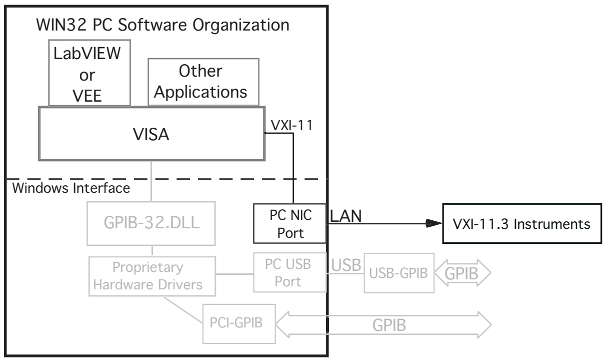

Figure 5B Controlling VXI-11.3 Instruments through VISA

Figure 5B shows a similar path through VISA to a VXI-11.3 compatible instrument. In Figure 5A, the VISA resource string specifies *gpib0, device address* to select a GPIB instrument. In Figure 5B, the VISA resource string specifies *instN* to select an instrument. In both cases, the VISA library handles all of the VXI-11 RPC protocol.

VXI-11 PACKETS and RPC

VXI-11 was designed to function over RPC. RPC stands for Remote Procedure Call and is a protocol designed by Sun Microsystems many years ago (See Reference 8). The VXI-11 messages listed in Table 1 are encapsulated within RPC messages. An RPC message contains the function number and all of the arguments required by the RPC service such as VXI-11. Each RPC message from the client is acknowledged by a response message from the service.

RPCGEN CODE GENERATOR

As noted earlier, the VXI-11 protocol is based on Remote Procedure Calls (RPC). The VXI-11 Specification includes a listing of the RPCL function prototypes. If RPC Capability has been installed on your computer, then you have utility called rpcgen that can be used to install the RPC stub functions in your operating system.

The rpcgen utility converts the RPCL function prototypes found in the VXI-11 Specification into the four files shown in Figure 6. These files are tailored specifically for your operating system and computer system by the rpcgen utility. If you change computers or update the operating system, you only need to run the rpcgen utility to get a set of updated files.

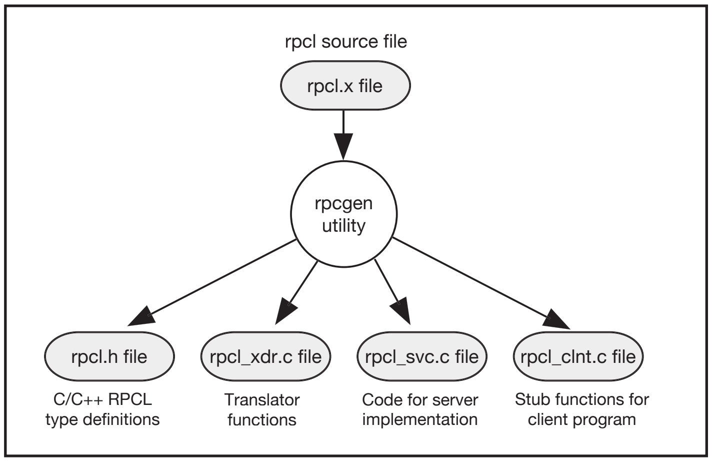

Figure 6 RPCGen Output Files

The .h file is a C header file. It contains ready-to-use C declarations of the numbers and data types in the VXI-11 RPCL function list. The _svc.c file can be ignored unless you are implementing a server. Note that a server implementation is only required if you are going to use an interrupt channel.

The _clint.c file contains the stub functions for your application program to call.

USING THE BASIC RPC FUNCTIONS

Table 2 VXI-11 Program Steps

Step	Function	Description
1	clnt_create()	Creates RPC client and opens a channel to the RPC server on a VXI-11 device.
2	create_link()	Establishes a link to a logical device
3	device_write()	Sends a message or command to the logical device.
4	device_read()	Reads a response message from a logical device.
5	destroy_link()	Closes the link to the device.
6	clnt_destroy()	Destroys the RPC client and closes the channel.

Table 2 lists the basic steps needed to program a VXI-11 device. It starts with using the clnt_create() function to create a RPC client and the core channel to the device. Next create a link to the device with the create_link() function. The returned data structure includes a link ID number that is used for subsequent calls RPC calls to the device. It also includes a parameter that specifies the size of the device's input buffer. Repeat steps 1 and 2 for each remaining logical device until channels and links have been created to all of the VXI-11 instruments that the program will be controlling.

Next, the device_write function is used to send commands to the device. Because VXI-11 devices are IEEE-488.2 devices, all of the IEEE 488.2 protocol and format rules apply to the commands sent to a VXI-11 device. This means that you program them the same way you would program a modern GPIB instrument. The response data is not a response to a query but rather an acknowledgement that the message was received by the device. Any transmission or command errors will be indicated in the response packet.

Use the device_read() function to read data or query responses from the device. The termChar parameter defines the character used by the device to terminate its response message.

The device_write() and device_read () functions are used as required in the program to control the devices.

At the end of the program, use the destroy_link() function to release the link to each device linked to at the beginning of the program. Then use the clnt_destroy() function to close the channel and release all of the instrument resources.

Some VXI-11 program examples show steps 2, 3, 4 and 5 together which implies that all four steps are all required to communicate with a VXI-11 instrument. The result is that some programmers put unnecessary create_link() and destroy_link() functions in their program each time they communicate with a VXI-11 instrument. Only open a link to a logical device at the beginning of the program and keep it open until done with the instrument. This is good programming practise and will make your programs run faster. Note that some vendors' instruments will 'abort' the channel if the link count goes to zero. This is another good reason to maintain the link until the end of the program.

Likewise do not repeatedly open and close channels. Create the channel(s) to the device(s) at the beginning of the program and leave them open until the end of the program. Channel opening is a time consuming operation for the application program

Note that if locks are being used to keep an instrument from being accessed by another client, they should be released as soon as the instrument can be shared. If the physical connection to the instrument is accidentally broken, the instrument will remain locked until it can determine the link is broken or until it is reset or power cycled. If the original client tries to relink to the device, it will be blocked because the lock is still active and belongs to the broken link.

TIMEOUTS

ICS VXI-11 devices include an internal COMM_timeout which is the period of time the device will go without a message from the client before terminating any lock and destroying the link. Set COMM_timeout to 2-5 minutes when debugging a new program and the program is likely to be abruptly terminated without closing the links and channel. This way a devices limited resources are released sooner. After the program has been debugged, the COMM_timeout period can be set to a longer time. If the physical link is subject to failures, it is best to make the COMM_timeout period shorter so the device's resources will be available sooner. Wire based systems are fairly dependable so the COMM_timeout period can be hours long. It is a good idea to add a background function to application programs that periodically performs some non-invasive VXI-11 function on each device that the program is linked to if the program has not sent that device a message in a certain amount of time. The time should be less than each device's COMM_timeout period. This will prevent unintentional channel closures when the application is left idle for long periods of time.

Another LAN device element is KeepAlive. KeepAlive is a TCP/IP function that enables the device's socket layer to send the client a short test message once every 120 minutes. If the client fails to reply, the device will close the socket, release all locks and reclaim all associated instrument links. If you do not use KeepAlive or a COMM_timeout mechanism, then the device can run out of resources and be inaccessible if it does not have a way to recover unused resource. Do not enable Keep Alive if the network or the client does not support Keep Alive messages.

ABORT CHANNEL

The Abort channel is used to send the device_abort() RPC to the logical device to terminate the current operation. The Abort channel requires its own clnt_create() and create_link() function calls to set the channel up. When setting up the Abort channel, note that it does not register itself with the port mapper service. The TCP port number used by the RPC server needs to be explicitly stated when calling the clnt_create() function. The Abort TCP port number was returned as part of the Core channel returned data structure. Only the device_abort() RPC can be sent over the Abort channel.

SERVICE REQUESTS (SRQs)

Service Requests in a VXI-11 device are setup just as they would be in a similar GPIB instrument. In the VXI-11 world, Service Requests are passed back to the host through a reverse Interrupt channel. Here the device acts as a client and sends the device_intr_srq() RPC to a server application running in the host. The job of the server application is to receive the request and inform the client application that a device needs service. Typically this is done by setting a flag that the client can periodically examine.

Use the functions in the _svc.c file to setup the server application. The server application typically runs as a separate thread. Use a svc_run() call to start the RPC server. Note that this call does not return.

It should be mentioned that a reverse Interrupt channel has problems because there are no periodic messages sent over it. It is possible for the reverse channel to die without any notification. If this occurs, the application is left without a way to get a Service Request at a crucial time. A better approach is to periodically test the device's status in the client application.

VXI-11 AND LXI

VXI-11 is a protocol defining specification whose goal is to control IEEE-488.2 compliant instruments over a TCP/IP network. LXI is a recent specification for Ethernet instruments that stresses timing, packaging and uniformity of the user experience. Unfortunately the LXI Specification does not specify a communication protocol but it requires that LXI instruments use the undefined 'VXI Discovery

Method'. It is also does not require that LXI instruments be IEEE-488.2 compliant but only that they must respond to an "*IDN?" query. Buyers of LXI instruments need to be aware of these shortcomings in the LXI Specification and, when buying a LXI instrument, should check the proposed instrument's specifications to be sure that it is VXI-11.3 and IEEE-488.2 compliant. Else, the user may be restricted to controlling the instrument through the vendor supplied IVI drivers and any other supplied programming methods. Refer to ICS Application Bulletin AB80-10 for more information about the differences between the VXI-11 and LXI Specifications.

SUMMARY

This application note has described the VXI-11 Specification, its command set, the use of Remote Procedure Calls (RPC) and how the client application links to devices. Guidelines are also included on when to use a VISA library and when to use the rpcgen utility to prepare files tailored to your computer. Some information is included about RPC programing but the user should refer to ICS's other AB80 series Application Bulletins for a more detailed description about using RPC. Many additional RPC programming references are also available on the Internet if you search for 'RPC Programming'.

The VXI-11 Protocol provides a rugged means of connecting a client application to a set of instruments over a TCP/IP network. Other communication methods may have improved performance but they do not offer the same ease of portability among different operating systems or the error free performance of the VXI-11 protocol. They also require special drivers that limits their application to a specific operating system.

REFERENCES

The following references are reprinted from the VXI-11 Specification:

[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] Internet Protocol, Request for Comments 791, Jon B. Postel, DDN Network Information Center, SRI International, September, 1981, See also MIL-STD 1777.
[4] Transmission Control Protocol, Request for Comments 793, Jon B. Postel, DDN Network Information Center, SRI International, September, 1981, See also MIL-STD 1777.
[5] A Standard for the Transmission of IP Datagrams over Ethernet Networks, Request for Comments 894, C. Hornig, DDN Network Information Center, SRI International, April 1984.
[6] XDR: External Data Representation Standard, Request for Comments 1014, Sun Microsystems, DDN Network Information Center, SRI International, June, 1987.
[7] A Standard for the Transmission of IP Datagrams over IEEE 802 Networks, Request for Comments 1042, J. Postel and J. Reynolds, DDN Network Information Center, SRI International, February 1988.
[8] RPC: Remote Procedure Call Protocol Specification, Request for Comments 1057, Sun Microsystems, DDN Network Information Center, SRI International, June, 1988.
[9] Requirements for Internet Hosts -- Communication Layers, Request for Comments 1122, R. Braden, DDN Network Information Center, SRI International, October, 1989.
[10] ISO 8802-2:1989[ANSI/IEEE 802.2-1989] Information Technology - Local and Metropolitan Area Networks - Part 2: Logical Link Control.
[11] ISO/IEC 8802-3:1993 [ANSI/IEEE 802.3-1993] Information Technology - Local and Metropolitan Area Networks - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.
[12] The Ethernet, Physical and Data Link Layer Specifications, Version 2.0, Digital Equipment Corporation, Intel Corporation, and Xerox Corporation, 1982.