ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

RUNNING AN ICS 80xx INSTRUMENT INTERFACE FROM THE APPLE OS X OPERATING SYSTEM

INTRODUCTION

With the resurgence in the popularity of Apple's computers, people are using them more often in engineering applications. Since the Apple OS X operating system is a Unix based system, Apple computers are easy to use with any of ICS's instrument servers that use Remote Procedure Calls (RPC). This application bulletin shows how to quickly install the VXI-11 RPC libraries in an Apple computer and how to develop a simple C program. This application bulletin can be used with ICS's 8003, 8013, 8063, 8064 and 8099 Ethernet interfaces. For the rest of this application bulletin these interfaces will be referred to as just 80xx.

VXI-11 INTRODUCTION

ICS's 80xx Ethernet Interfaces are VXI-11 compliant servers. The VXI-11 Specification specifies a protocol for communication with devices over a network (LAN) via TCP/IP. This protocol uses the ONC/RPC (Open Network Computing/Remote Procedure Call) standard which is based on TCP/IP and is described in the 'VXI-11: TCP/IP Instrument Protocol Specification'. While the VXI-11 Specification was written for test and measurement applications, it can be used in many other applications.

VXI-11 is the overall VXI-11 document and describes the network protocol. There are three sub-specifications. VXI-11.1 is for a VXI chassis and is not applicable to the 80xx. The VXI-11.2 Specification defines how a gateway (like ICS's Model 8065) controls the GPIB bus and performs bus specific functions. The VXI-11.3 Specification describes instrument specific functions including data transfer to and from an instrument like ICS's 8063.

ICS's 80xx Ethernet interfaces have VXI-11.3 capability so they can be used just as you would use any GPIB instrument with IEEE-488.2 capability.

To learn more about the VXI-11 protocol, download our VXI-11 Tutorial, AB80-11, from our website at http://www.icselect.com/ab_note.html#VXI11

PREPARATION

The first step is the conversion of the RPCL in the VXI-11 Specification to the RPC files specific to your Apple computer. Download a copy of the VXI-11 Specification from ICS's website (http://www.icselect.com/vxi_spec.html) or from the VXIbus Consortium's website (www.vxibus.org). Copy the RPCL in the VXI-11 Specification to a vxi11.x file on your MAC.

Install Apple's Developer Tools on your MAC. You can obtain the Developer Tools from Apple's website.

CREATING THE RPC FILES

Perform the following steps to generate the RPC files. Apple's OS X 10.5 and 10.6 includes an rpcgen utility that does the RPC file generation.

1. Open a Terminal Session.
2. Navigate to the folder with the vxi11.x file.
3. Type rpcgen vxi11.x

This will generate the following files: vxi11_clnt.c, vxi11_svc.c, vxi11_xdr.c and vxi11.h. You will not use the vxi11_svc.c file in this client application.

DEVELOPING THE PROGRAM

1. In Xcode (part of the Apple Developer Tools) create a new project (Command Line utility > Standard Tool).
2. Add the vxi11_clnt.c, vxi11_xdr.c and vxi11.h files to the project.
3. Write your main.c file.

RPC programming requires that you first create a link to a device before you can send it commands or read data back from it. When done with the program you destroy the link before exiting the program. The recommendation for the 80xx interfaces is that you create a link to the 80xx for each inst personality that you will be using as part of the initialization phase of your program. These links should remain open until you exit the program.

Inst personalities are similar in concept to a GPIB device with multiple addresses. *inst0* is analogous to a GPIB device's primary or main address. *inst1* and higher personalities are analogous to a GPIB device's dual primary or multiple secondary addresses.

Simple program examples that have an open-write-read-close sequence often confuse new users since they are lead to believe that open-write-read-close is the normal sequence for all commands and queries. This is a problem with a short example whicvh should not be followed. Opening and closing links is very time intensive and causes the 80xx interfaces to quickly run out of resources.

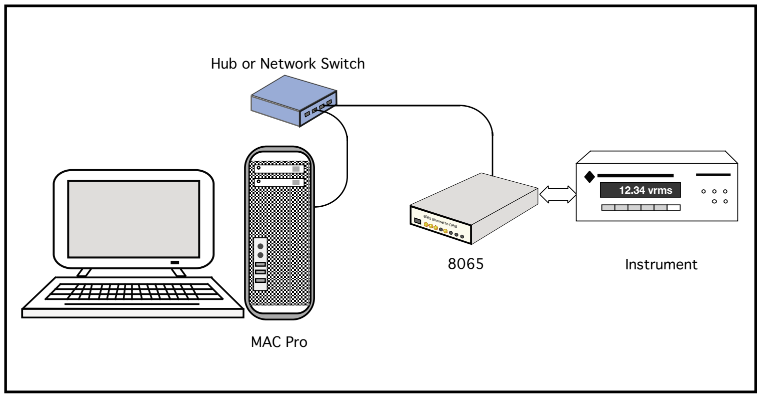

Figure 1 Example Test Setup

EXAMPLE PROGRAM

The following example was adapted from one written by Bela Farago from Institute Laue-Langevin in Grenoble, France. The program is a simple one that does an IDN query of the instrument. It has been adapted here to query an 80xx interface.

The setup is shown in Figure 1. The interface used is an ICS Model 8063. The program includes calls to Microseconds() to time the read, write and create_link operations. These calls should be removed from your actual program. The execution times Bela measured with an Intel core-duo mac are:

create_link - 265 ms
write-read - 65 ms
close link and destroy client - 20 ms

The measured times reenforces our recommendation to create the links to all of your instruments at the start of your program and to leave them open until you end the program. Link creation and destruction are major time sinks.

EXAMPLE MAIN.C

```
#include <stdio.h>
#include <rpc/rpc.h>
#include <string.h>
#include <CoreServices/CoreServices.h>
#include “vxi11.h”

#define WAITLOCK_FLAG 1
#define WRITE_END_CHAR 8
#define READ_END_CHAR 0x80
#define VXI_ENDW (WAITLOCK_FLAG | WRITE_END_
        CHAR)

CLIENT *rpcClient;
Create_LinkParms crlp;
Create_LinkResp *crlr;
Device_WriteParms dwrp;
Device_WriteResp *dwrr;
Device_ReadParms drdp;
Device_ReadResp *drdr;
Device_Error *derr;

UnsignedWide microTickCount1,microTickCount2;

static char *svName = “192.168.1.254”;

int main(){
char sendMessage[1024];
//char responseMessage[2048];

        rpcClient = clnt_create(svName, DEVICE_CORE,
                DEVICE_CORE_VERSION, “tcp”);
        if (rpcClient == NULL){
                printf(“Error creating client\n”);
                clnt_pcreateerror(svName);
                return 1;
        }
//      printf(“Hello %s\n”, svName);
        crlp.clientId = (long) rpcClient;
        crlp.lockDevice = 0;
        crlp.lock_timeout = 10000;
        crlp.device = “inst0”;
        crlr = create_link_1(&crlp, rpcClient);
        if (crlr == NULL){
                clnt_perror(rpcClient, svName);
                return 1;
        }
//      printf(“Link created to %s\n”, crlp.device);

        Microseconds ( &microTickCount1);

        dwrp.lid = crlr->lid;
        dwrp.io_timeout = 1000;
        dwrp.lock_timeout = 10000;
        dwrp.flags = VXI_ENDW;
        sprintf(sendMessage, “*IDN?\n”);
        dwrp.data.data_len = strlen(sendMessage);
        dwrp.data.data_val = sendMessage;
        dwrr = device_write_1(&dwrp, rpcClient);
        if (dwrr == NULL){
                clnt_perror(rpcClient, svName);
                printf (“device_write returned dwrr ==
                        NULL \n”);
                return 1;
        }
        drdp.lid = crlr->lid;
        drdp.io_timeout = 1000;
        drdp.lock_timeout = 10000;
        drdp.flags = VXI_ENDW;
        drdp.termChar = ‘\n’;
        drdp.requestSize = 1024;
        drdr = device_read_1(&drdp, rpcClient);
        if (drdr == NULL){
                clnt_perror(rpcClient, svName);
                printf(“device read returned drdr == NULL\n”);
                return 1;
        }
//      printf(“response = %s\n”, drdr->data.data_val);

        Microseconds ( &microTickCount2);


        derr = destroy_link_1(&(crlr->lid), rpcClient);
        clnt_destroy(rpcClient);
        printf(“Ellapsed time in microsec %ld \n”, microTick-
                Count2.lo-microTickCount1.lo);
        return 0;
}
```

SUMMARY

This Application Note has shown how ICS's 80xx series Ethernet Interfaces can be controlled from an Apple computer using a program developed in Apple's Xcode. The MAC RPC file generation is easily done using the OS X's built-in rpcgen utility to convert the VXI-11 RPCL into MAC specific files.

The Application Note also includes an Xcode example that does the IEEE-488.2 *IDN query.

CREDITS

Special thanks to Bela Farago at Institut Laue-Langevin, BP:156, 38042 Grenoble Cedex 9, France for the original program example and comments about using the 8065 with a MAC.

References:

The following references were used in the course of developing this program:

* Model 8063 Ethernet <-> Parallel Interface Instruction Manual, ICS Electronics.

* Frequently Asked Questions (FAQ) For The ICS Model 8065 Ethernet-To-GPIB Controller, Application Note AB80-1, ICS Electronics.

* VXI-11 RPC Programming Guide for the 8065: An Introduction to RPC Programming, Application Note AB80-3, ICS Electronics.

* VMEbus Extensions for Instrumentation: TCP/IP Instrument Protocol Specification VXI-11, Revision 1.0, July 17, 1995.

* Power Programming with RPC, John Bloomer, O’Reilly & Associates, Inc., 1992.