VXI I/O Using the IEEE 1394 Serial Bus (FireWire)

FireWire, IEEE 1394, IEC 1883. These titles refer to a high-speed serial bus that is literally a new standard for transmitting data between PCs and consumer electronics. FireWire, as named by its inventors at Apple Computer Inc., is designed for applications where large amounts of digital audio and video data are recorded, edited, stored, and transferred between devices. The bus's performance, flexibility, and ease of use have caught the attention of engineers at Hewlett-Packard's Measurement Systems Division (MXD). As a result, IEEE 1394 has been implemented as an I/O interface between external PCs and C-size VXI mainframes.

This paper describes the adaptation of IEEE 1394 to VXI systems from hardware, software, and programming perspectives.

IEEE 1394 Topology and Terminology

The best place to begin the description of IEEE 1394 is with an overview of its topology and definitions of the terms associated with its use. Figure 1 shows a VXI system consisting of a PC and three VXI mainframes - interconnected with the IEEE 1394 bus.

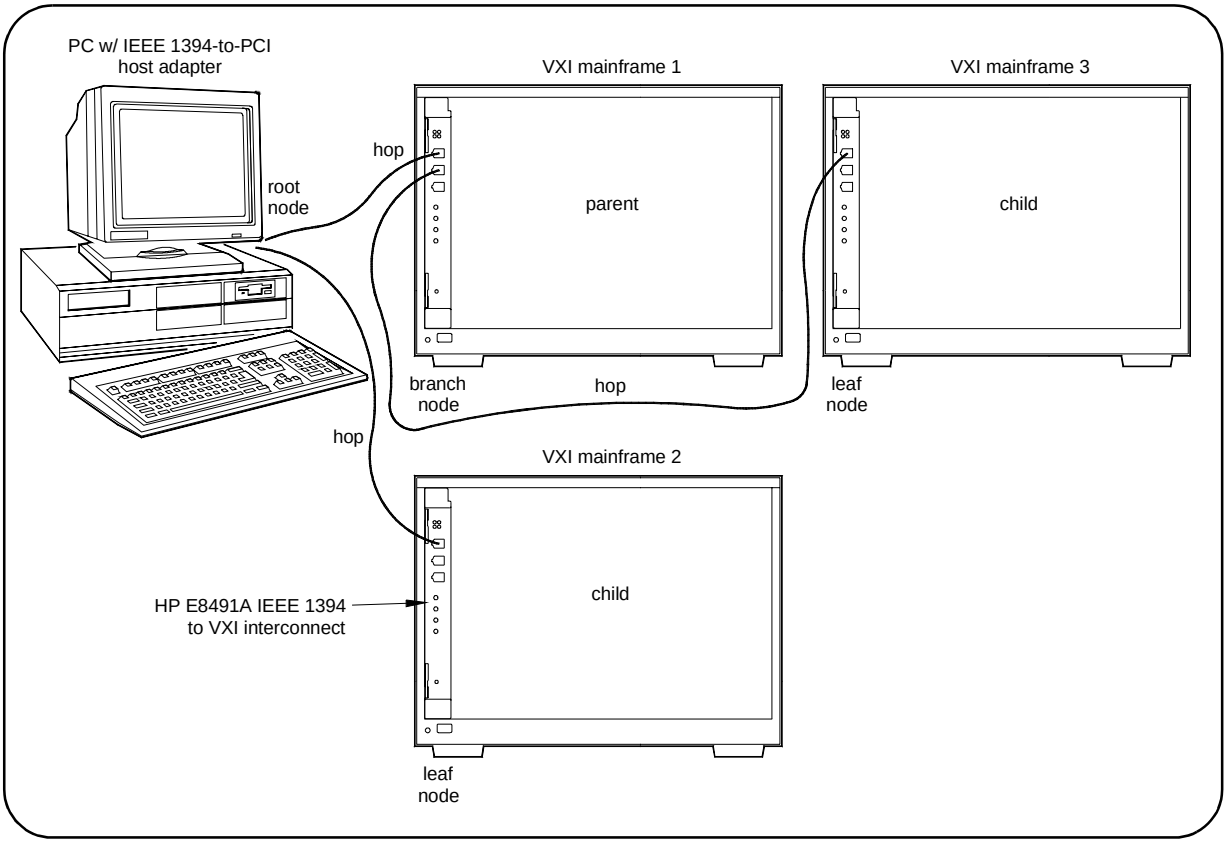


Figure 1. IEEE 1394 Topology and Terms.

The terms shown in Figure 1 are defined in the following table.

Table 1. Definition of Terms.

Host Adapter	Links the computer's PCI bus to the IEEE 1394 interface. To use a host adapter, computers must be PCI Rev. 2.0 compliant.
HP E8491A 1394-to-VXI Interconnect	Links the IEEE 1394 interface to the VXI backplane. Provides the backplane's clock and trigger resources.
root node	Each device (HP E8491A) on the bus is a "node." In VXI systems, the PC is always the root node having cycle master and bus master capabilities.
branch node	A branch node has IEEE 1394 cables connected to two or more ports. In Figure 1, VXI mainframe 1 is a branch node because of the 1394 cables connecting it to the PC (root node) and to VXI mainframe 3 on its right.
leaf node	A leaf node has a single IEEE 1394 cable connected to it.VXI mainframes 2 and 3 are leaf nodes.
parent	A node (HP E8491A) is a parent if it is physically connected closer to the root than an adjacent node. In Figure 1, VXI mainframe 1 is a parent node because it is closer to the root than VXI mainframe 3.
child	A node (HP E8491A) is a child if it is farther from the root than an adjacent node. In Figure 1, VXI mainframe 3 is a child node because it is farther from the root than VXI mainframe 1. A node with a single IEEE 1394 cable connected to it (leaf node) is always a child (VXI mainframe 2).
hop	A hop is a IEEE 1394 cable link between nodes. There can be no more than 16 hops between any two nodes. In the diagram above, there is a maximum of three hops between nodes. The distance between any two nodes cannot exceed 72m.

Features of the IEEE 1394 Bus

The following features of the IEEE 1394 bus apply to all bus applications including VXI systems.

* Daisy-chain or branching configurations are allowed. There can be no closed loops (i.e. more than one connection between any two devices).

* Up to 63 devices (including 16 HP E8491As) are allowed per bus segment. One host adapter represents one bus segment.

* There is a maximum of 16 hops between any two devices. The bus cable length cannot exceed 72 meters between any two devices.

* Live/hot connections. A VXI mainframe anywhere in the configuration can be turned on/off at any time without affecting the other mainframes. The IEEE 1394 bus automatically reconfigures itself any time a VXI mainframe (or other device) is added or removed.

Optimizing the Configuration

I/O performance is impacted slightly by the hardware configuration.The VXI mainframe closest to the PC (root node) has the highest priority. For example, if instruments in VXI mainframes 1 and 3 (Figure 1) contend for the bus at the same time, the root node will grant mainframe 1 access to the bus first. However, the bus's fair arbitration protocol (covered in the next section) ensures that each device has equal access to the bus, and that devices closer to the root are not continually granted the bus.

Data Transfer Protocol

Data transfer over the IEEE 1394 bus can be either asynchronous or isochronous[1]. Hewlett-Packard's IEEE 1394 based VXI systems use asynchronous data transfers and a "fair arbitration" protocol to ensure each VXI mainframe has equal access to the bus. Figure 2 illustrates the concepts of asynchronous data transfers and fair arbitration.

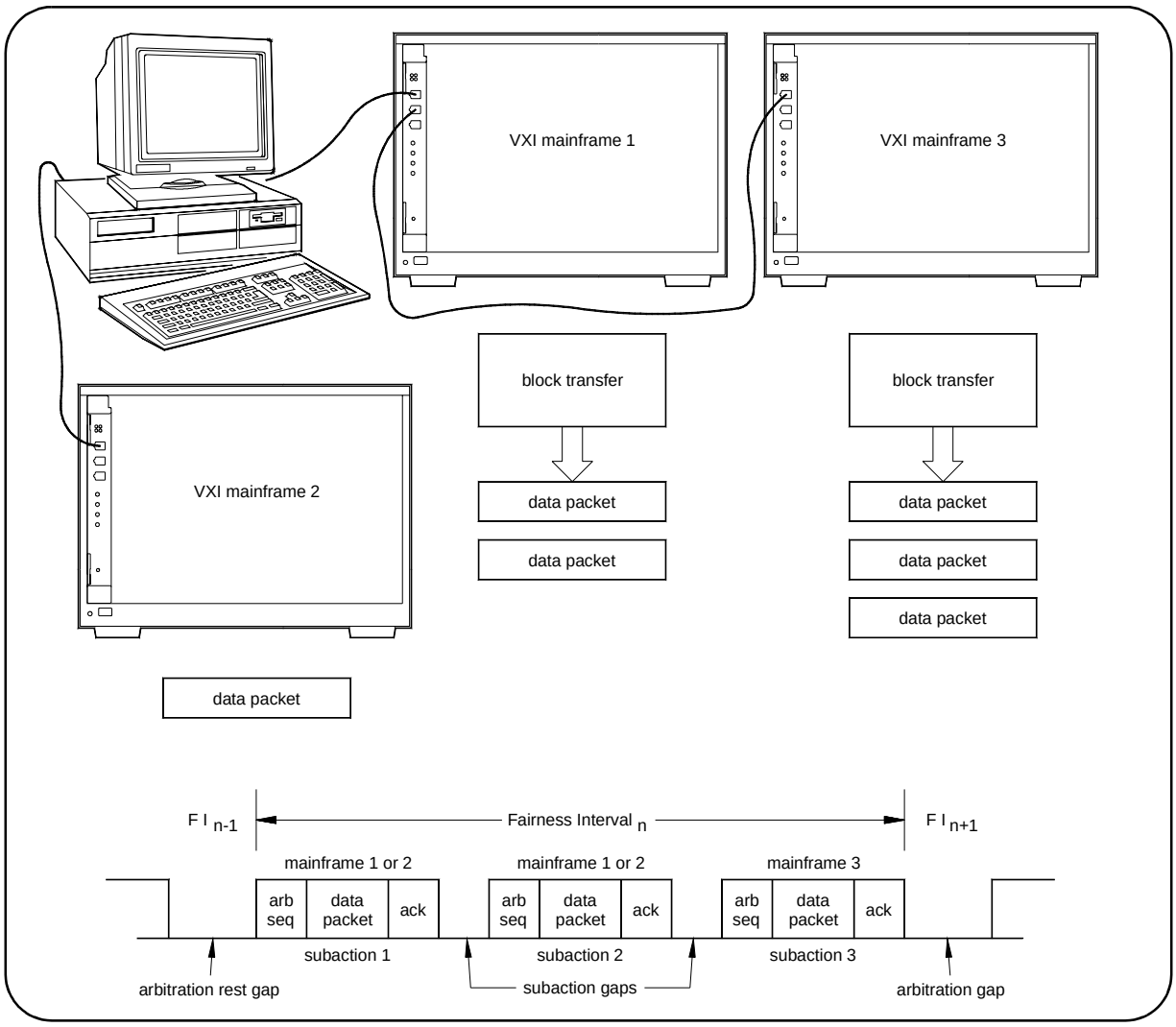


Figure 2. IEEE 1394 Data Transfer Protocol.

1. Isochronous data transfers broadcast variable amounts of data at regular intervals with no acknowledgement. Isochronous and Asynchronous data transfers can occur on the same bus.

Asynchronous Data Transfers

During an asynchronous data transfer, a variable amount of data is transferred to an explicit address in real time, and an acknowledgement is returned. Data is transferred across the IEEE 1394 bus in packets called "subactions." An asynchronous subaction is made up of three parts:

* arbitration sequence - the period when a device requests control of the bus in order to transmit a data packet.

* data packet - the data packet consists of a data prefix that contains information about the transaction, the data itself (e.g. VXI instrument commands), and a data end signal. The maximum packet size is 1 kByte for 200 Mbit host adapters and 2 kBytes for 400 Mbit adapters.

* acknowledgement - a code returned by the (addressed) data destination indicating the action taken by the receiver.

The periods between subactions are called subaction gaps. The subaction gap allows devices that have not had control of the bus during the current "fairness interval" to arbitrate for control.

Fair Arbitration Protocol

The fair arbitration protocol is based on the fairness interval shown in Figure 2. A fairness interval consists of one or more subactions in which data packets are transferred over the bus. A fairness interval is as follows:

1. The interval begins when devices (HP E8491A's) arbitrate for control of the bus.

2. When a device is granted control, it transfers its data packet and is then disabled from arbitrating until the next fairness interval.

3. A subaction gap occurs after the previous data packet is transferred. During this period, remaining devices arbitrate for the bus. The next device granted the bus transfers its data packet and is then disabled from arbitrating until the next fairness interval.

4. The fairness interval ends after each device has had an opportunity to access to the bus and the arbitration reset gap, which is longer than the subaction gap, occurs. The arbitration reset gap re-enables each device for arbitration during the next fairness interval.

VXI Data Transfers

To take advantage of the IEEE 1394 data transfer protocol, large amounts of data should be transferred between VXI instruments and the PC using block transfers. During a block transfer, data is divided into the packets described above; the number of packets depends on the amount of data and whether a 200 Mbit or 400 Mbit host adapter is used. Compared to protocols that transfer data one byte or one word at a time, transfer speed between the instrument and the PC is increased because the IEEE 1394 protocol overhead is associated with the fairness interval and with each packet, rather than with each byte or word transferred. Thus, transfer speeds (bits/second) over the IEEE 1394 bus increase as the amount of data transferred (block size) increases.

The "Programming and Performance" section of this paper contains more information on VXI block transfers and includes a program that benchmarks transfer speeds over various interfaces.

Implementing IEEE 1394 in a VXI System

The components of an IEEE 1394 - based VXI system are listed in the table below.

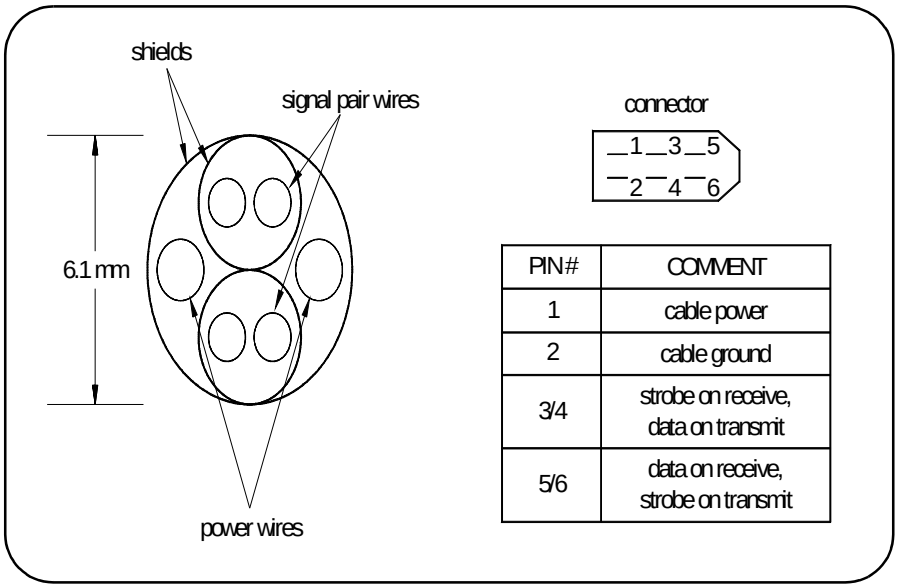
Table 2. Components of an IEEE 1394 - Based VXI System.

Hardware	Software	Procedure
IEEE 1394-to-PCI Host Adapter	HP I_O Libraries	1. Install host adapter into PC. 2. Install HP E8491A in slot 0 of VXI mainframe. 3. Install the HP I_O Libraries. 4. Configure the interface using I_O Config from the I_O Libraries. 5. Install VXIplug&play drivers. 6. Install instruments in the VXI mainframe.
HP E8491A PC Link-to-VXI Interconnect w/cable		
VXI Instruments	HP Universal Instrument Drivers	

The IEEE 1394 Host Adapter and Interface Cable

The IEEE 1394-to-PCI host adapter is a PC plug-in card capable of transferring data at up to 400 MBits/second. Each host adapter card represents one bus segment and can support up to 63 nodes (including 16 HP E8491As). If necessary, a host adapter can supply 8V to 40V DC at up to 1.5A. HP's implementation of IEEE 1394 presently supports one host adapter.

The interface cable supplied with the host adapter has two power wires and two signal twisted-pairs. A cross-section of the cable and the cable connector are shown below:



PIN#	COMMENT
1	cable power
2	cable ground
3/4	strobe on receive, data on transmit
5/6	data on receive, strobe on transmit

Figure 3. Cross-section of the IEEE 1394 Cable.

The power wires route power from the host adapter to devices (nodes) on the bus, whether the devices are turned on or off. Since each device in the system acts as a repeater, the power supplied to a device that is turned off enables signals to be transferred across that device. This maintains signal continuity throughout the system.

The HP E8491A IEEE 1394 to VXI Interconnect

The HP E8491A IEEE 1394 interconnect links the VXI backplane to the IEEE 1394 bus. However, the E8491A and the IEEE 1394 bus do not extend the (VXI) backplane analog bus, digital bus, or Local bus between frames in multi-frame VXI systems. This means that the multimeter and multiplexers in a VXI scanning multimeter for example, must be installed in the same mainframe. Devices sharing the Local bus must also be installed in the same mainframe.

The E8491A is a VXI C-size device (Figure 4) normally installed in slot 0 of the mainframe. With a logical address of 0, the E8491A functions as the mainframe's resource manager via software included with the I_O libraries. The E8491A has 128 kBytes of shared RAM and contains many of the clock and triggering features found on the HP E1406A Command Module - a VXI resource manager/slot 0 device common in many GPIB-based systems. Unlike the E1406A, a VXI mainframe with the HP E8491A in slot 0 can be powered on/off at any time without affecting other mainframes in the system.

Using the HP E8491A with the HP E1406 Command Module

Though not a common configuration, the HP E1406 command module can be used in the same mainframe as the HP E8491 to provide GPIB access to instruments. For this configuration, however, the E8491A must be the mainframe's resource manager and slot 0 device.

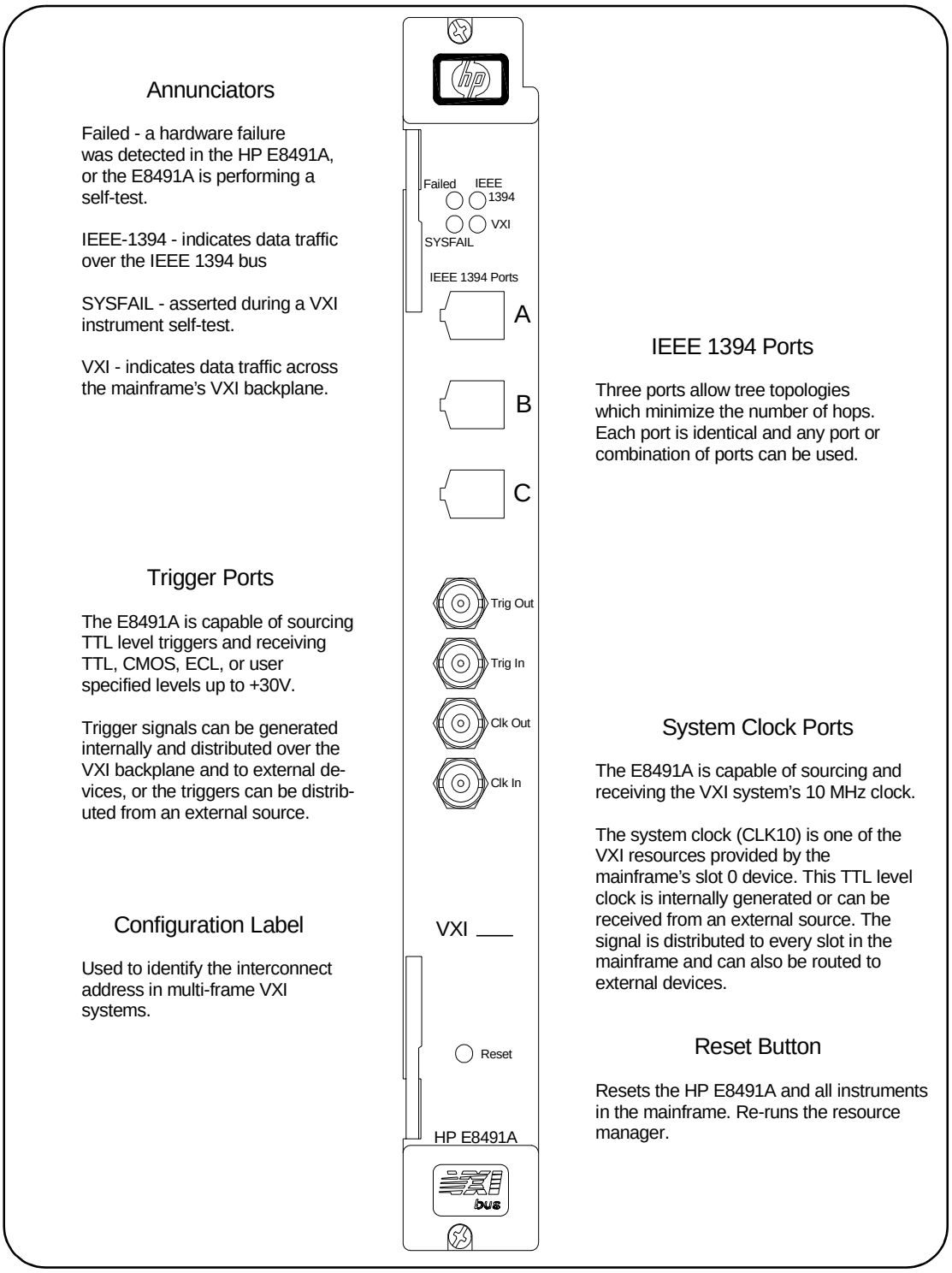


Figure 4. The HP E8491A IEEE 1394 to VXI Interconnect.

The HP I_O Libraries

The software required to use the IEEE 1394 interface in a VXI system is contained in the HP I_O Libraries and HP VXI*plug&play* Drivers. The software supports the Windows 95 and Windows NT platforms.

The software "stack" shown below shows the relationship of the VXI*plug&play* drivers to HP VISA/ HP SICL, to the host adapter drivers, and to the VXI instruments. Notice that Hewlett-Packard's implementation of the IEEE 1394 interface requires HP VISA and will not work with the VISA supplied by other vendors.

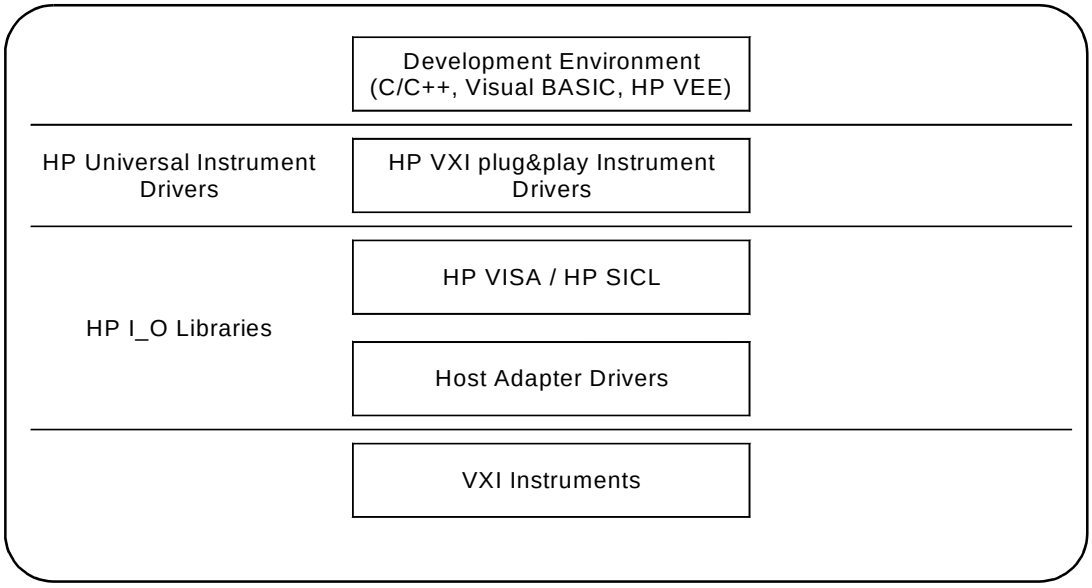


Figure 5. System Software and Drivers.

Configuring the HP E8491A Interconnect

Configuration of the E8491A is done with the HP I_O Libraries' 'I_O Config' utility. Running I_O Config brings up the following window with the E8491A listed (if present) as shown.

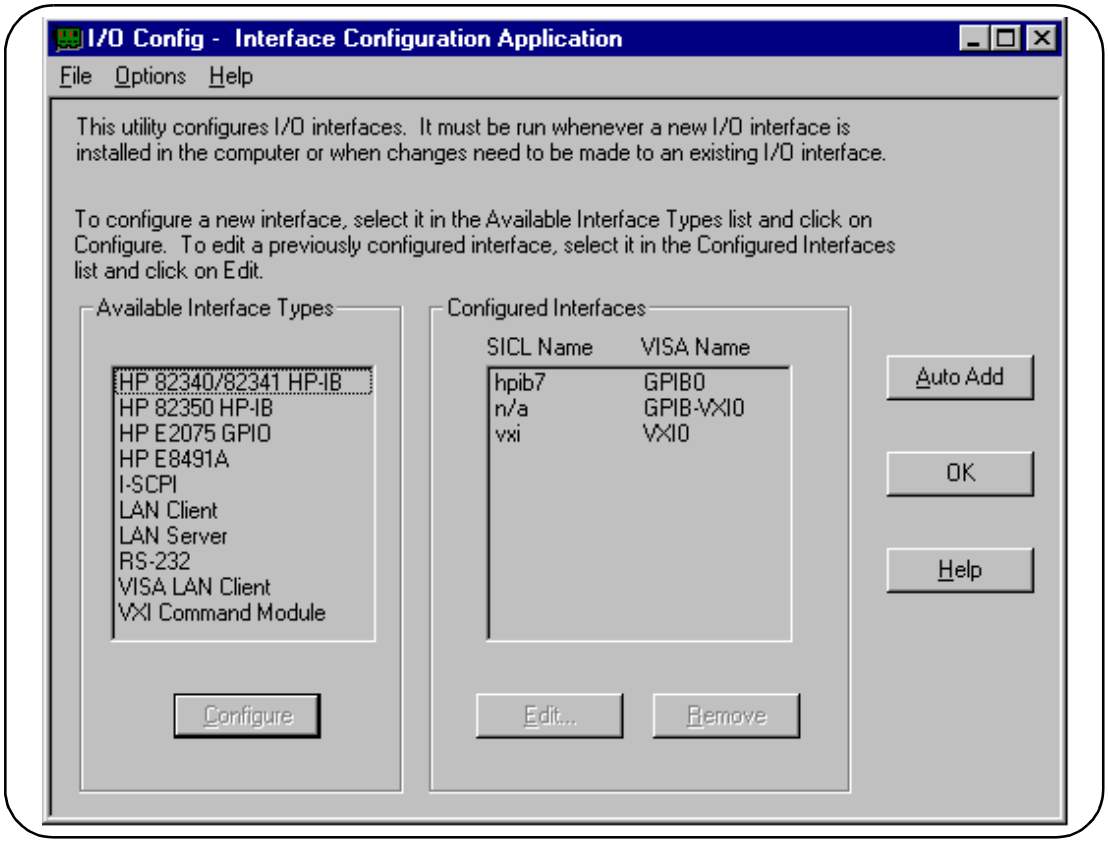


Figure 6. The HP I_O Libraries 'I_O Config' Utility.

The interface is configured or edited by selecting E8491A or VXI0 and then clicking on Configure or Edit. This brings up the configuration window shown in Figure 7.

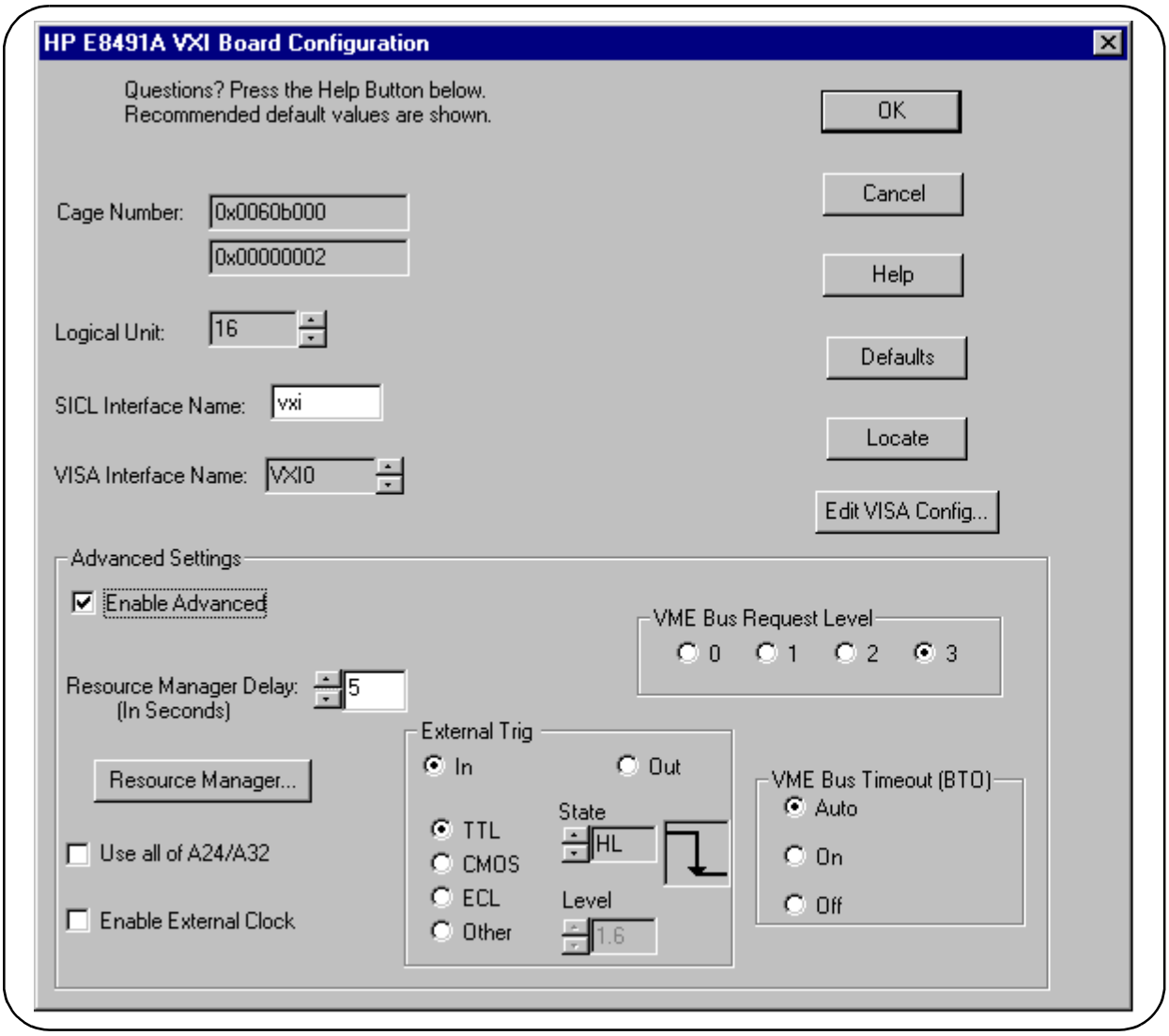


Figure 7. Configuring the HP E8491A Interconnect.

This window reflects the VXI resource manager and slot 0 capabilities available through the HP E8491A. Additionally, the window sets the (VISA/SICL) interface name which uniquely identifies each HP E8491A and which is used to address each instrument in the mainframe.

Programming and Performance

From a programming perspective, the IEEE 1394 bus is a transparent connection between the PC and the VXI mainframe. Applications developed using HP VEE, HP VXI*plug&play* drivers, HP VISA, and SICL; running on GPIB, VXI embedded controller, and National Instrument's MXI-2 interfaces are easily ported to the IEEE 1394 interface. The benchmark program in this paper, for example, ran on the IEEE 1394, VXI embedded controller, and MXI-2 interfaces without any program modifications. Porting the program to the GPIB interface required using a different interface name, plus minor modifications unique to the program.

Best Practices

As mentioned in the Data Transfer Protocol section, information is passed between the computer and VXI mainframe in data packets. Throughput, therefore, is increased using block transfers.

Utilizing Block Transfers

When using an instrument's VXI*plug&play* driver, you will need to consult the driver help file to determine those functions that perform block transfers. For VISA programs, block transfers are achieved with the functions:

viMoveIn8 viMoveOut8
viMoveIn16 viMoveOut16
viMoveIn32 viMoveOut32

The benchmark program in this paper compares transfer rates for various interfaces using a VXI*plug&play* block transfer function and the HP VISA viMoveIn16 function.

SICL support of the IEEE 1394 interface introduces the following extended functions:

IMAPX
IUNMAPX
IDEREFPTR
IBLOCKMOVEX
IPEEKX8, IPEEKX16, IPEEKX32
IPOKEX8, IPOKEX16, IPOKEX32

These functions are described in the SICL documentation. An example program featuring the use of IBLOCKMOVEX can be found in the HP E8491A Configuration and User Guide.

Programming Register-Based and Message-Based VXI Instruments

Unlike the HP E1406 Command Module, there are no SCPI instrument drivers for register-based instruments installed in, or downloaded to the HP E8491A.While this does not impact message-based instruments, register based instruments in IEEE 1394-based systems must be programmed using VXI*plug&play* drivers from the HP Universal Instrument Drivers CD, or programmed using register-level peeks and pokes. Notice that register-level programming offers negligible performance gains over the VXI*plug&play* driver, yet dramatically increases program complexity.

Opening Instrument Sessions

Programming over the IEEE 1394 interface begins by opening a line of communication (instrument session) between the instrument and the driver or I_O library (VISA or SICL). An address that includes the interface name (see Configuring the HP E8491A Interconnect) and the instrument's logical address is used in opening these sessions.

Shown below are VXI program segments that open a session to the HP E1563 digitizer in VXI*plug&play*, HP VISA, and HP SICL programs.

HP VXI*plug&play*

```
ViSession vi;

// open device (VXIplug&play) session to the HP E1563
errStatus = hpe1563_init("VXI0::24::INSTR", VI_FALSE, VI_FALSE, &vi);
```

HP VISA

```
ViSession defaultRM, id;

//open device (VISA) session to the HP E1563
viOpenDefaultRM (&defaultRM);

viOpen (defaultRM, "VXI0::24::INSTR",VI_NULL,VI_NULL, &id);
```

HP SICL

```
INST id;

// open device (SICL) session to the HP E1563
id = iopen("vxi,24")
```

Or, to open a session to use the HP E8491A:

```
INST id;

// open device (SICL) session to the VXI interface
id = iopen("vxi")
```

The HP E8491A IEEE 1394 interconnect uses the interface name VXI (or vxi). The interface number (0 to 63) is assigned using the 'I_O Config' utility. The logical address of the HP E1563 digitizer is 24, and INSTR indicates a VISA instrument control resource .

The interface number is unique to each HP E8491A. This identifies each mainframe in multi-frame VXI systems and allows instruments with the same logical addresses to be installed in separate mainframes.

Benchmark Program

An important aspect of any I/O interface is performance. The following program compares data transfer rates and throughput between a VXI instrument and a PC using the IEEE 1394, GPIB, and VXI embedded PC interfaces.

The program configures the HP E1563A 2-Channel 800 kSa/s Digitizer and then takes 240,000 DC voltage readings. The readings are transferred from the digitizer's A24 space to the PC. Groups of 80,000 readings are transferred using three methods:

hpe1563_fetchAll_Q (HP VXIplug&play function)

viMoveIn16 (HP VISA function)

viIn16 (HP VISA function)

The first two methods (hpe1563_fetchAll_Q and viMoveIn16) are block transfers used for comparing transfer rates. The third method transfers the data one reading at a time and is used for comparing throughput.

The Results

The following table summarizes the results obtained from the various interfaces.

Table 3. Data Transfer Rate Comparisons.

Interface	hpe1563_fetchAll_Q*	viMoveIn16*	viIn16**
400 MHz IEEE 1394 host/ HP E8491A (200 MHz PC)	444,000 rdgs/sec	888,000 rdgs/sec	0.000200 seconds
HP 82341C HP-IB Interface card / HP E1406A Command Module (200 MHz PC)	16,400 rdgs/sec	33,300 rdgs/sec	0.000552 seconds
HP E6233 VXI Embedded Controller (166 MHz)	1,600,000 rdgs/sec	2,000,000 rdgs/sec	0.000013 seconds
* Block transfers of 2,000 readings per block. ** Used to determine the time required to request and receive a single reading.			

```
// BLOK1563.CPP - This program compares block transfer speeds using two methods:
// VXIplug&play function hpe1563_fetchAll_Q and VISA function viMoveIn16. The
// program also compares throughput using VISA function viIn16.

#include "hpe1563.h"// include the driver header file
#include "visa.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

// project files: BLOK1563.cpp, hpe1563.lib, VISA.lib

// Specify the addressing path. Note that GPIB-VXI addressing is through
// the HP E1406A Command Module.

#define E1563 "VXI0::64::INSTR"          // VXI addressing
//#define E1563 "GPIB-VXI0::64::INSTR"   // GP-IB addressing

// set up loops

#define T_LOOP 40      // number of loops to average
#define R_LOOP 2000    // number of readings per T_LOOP (viIn16)

// prototypes

void check(ViSession vi, ViStatus error);

void main(void)
{
  ViSession vi;
  ViSession defaultRM, id;
  ViStatus errStatus;
  ViInt16 rdgs[240000];
  ViInt16 *dataPtr;       // pointer to cast readings to 16-bit integers



  ViInt16 r, t;           // loop counters
  ViReal64 range;         // range variable for reading conversions
  long dataArrayLen=2000; // 2,000 readings each pass using hpe1562_fetch_All_Q
  ViInt32 numRdgs;
  ViChar err_message[256];

  DWORD start, stop, t1, t2, t3;    // timing variables

    dataPtr = rdgs;  // set pointer to rdgs array

    // open a VXIplug&play device session and reset the digitizer
    errStatus = hpe1563_init(E1563,0,1,&vi);
    if( VI_SUCCESS > errStatus)
    {
      hpe1563_error_message( vi, errStatus, err_message);
      printf("Unable to open %s\n", E1563);
      printf("hpe1563_init() returned error message %s\n", err_message);
      return;
    }

  // open VISA session to E1563 for I/O access over GPIB using
  // viMoveIn16 and viIn16
  viOpenDefaultRM (&defaultRM);
```

```
  viOpen(defaultRM, E1563, VI_NULL, VI_NULL, &id);

  // enable digitizer error detection
  hpe1563_errorQueryDetect(vi, 1);

  // set a 5s timeout period to allow functions to complete
  errStatus = hpe1563_timeOut(vi, 5000);
  check(vi, errStatus);

  // configure the digitizer to take 240,000 post-trigger readings
  // not to exceed 4V on channel 1
  errStatus=hpe1563_configure(vi, 1, 4.0, 240000,1);
  check(vi, errStatus);

    // set an immediate trigger
    errStatus = hpe1563_trigEvent(vi, 1, hpe1563_TRIG_IMM, 0.0);
    check(vi, errStatus);

    // set the minimum sample period
    errStatus = hpe1563_sampTim(vi, hpe1563_SAMP_TIM_MIN);
    check(vi, errStatus);

    // disable digitizer error detection
    hpe1563_errorQueryDetect(vi, 0);

   // initiate the digitizer
  errStatus = hpe1563_initImm(vi);

  // pause 315 ms to allow readings to complete
  Sleep (315);

  // transfer methods

  // begin timing transfer (hpe1563_fetch_All_Q)
  start = GetTickCount();

    for (t=0; t<T_LOOP; t++)
    {
      // fetch 1st 80,000 readings from A24 space, 2,000 readings each loop pass
      // (2,000 readings is the block limit for the hpe1563_fetchAll_Q function)
      errStatus = hpe1563_fetchAll_Q(vi, dataArrayLen, (ViInt32 *)rdgs, &numRdgs
);
    }

  stop = GetTickCount();

  t1 = stop - start;

  // begin timing transfer (VISA viMoveIn16)
  start = GetTickCount();

    for (t=0; t<T_LOOP; t++)
    {
      // transfer 2nd 80,000 readings from A24 space, 2,000 readings per block/pass
      // (change session name to 'id' when doing GPIB transfer)
      viMoveIn16(vi, VI_A24_SPACE, 0x2000, 2000, (ViPUInt16)
        &rdgs [80000+(t*2000)]);
    }

  stop = GetTickCount();
```

```
  t2 = stop - start;

  // set pointer to correct memory location and begin timing transfer (VISA viIn16)
  // (change session name to 'id' when doing GPIB transfer)
  dataPtr = &rdgs[160000];

  start = GetTickCount();

    for (t=0; t<T_LOOP; t++)
    {
      for (r=0; r<R_LOOP ;r++)
      {
        // transfer last 80,000 readings from A24, 1 reading per R_LOOP
        viIn16(vi,VI_A24_SPACE,0x2000, (ViPUInt16)(dataPtr++));
      }

    }

  stop = GetTickCount();

  t3 = stop - start;

  //print average reading transfer rates and throughput rate
  printf("Transfer rate using hpe1563_fetchAll_Q is %lf readings/second.\n\n",
(1000 * 2000 * T_LOOP)/(double) t1);

  printf("Transfer rate using viMoveIn16 is %lf readings/second.\n\n", (1000 * 2000
* T_LOOP)/(double) t2);

  printf("Round trip reading query speed is %lf seconds.\n\n", (double) (t3/1000)/
80000);

  // confirm readings transferred are valid
  dataPtr = (ViInt16 *)rdgs;

  // hpe1563_fetchAll_Q transfer
  errStatus = hpe1563_range_Q(vi, 1, &range);

  printf("Reading samples transferred using hpe1563_fetchAll_Q are: %lf %lf\n\n",
dataPtr[0]*range/32768, dataPtr[2]*range/32768);

  // viMoveIn16 transfer
  errStatus = hpe1563_range_Q(vi, 1, &range);

  printf("Reading samples transferred using viMoveIn16 are: %lf %lf\n\n",
rdgs[80000]*range/32768, rdgs[80002]*range/32768);

  // viIn16 transfer
  errStatus = hpe1563_range_Q(vi, 1, &range);

  printf("Reading samples transferred using viIn16 are: %lf %lf\n\n",
rdgs[160000]*range/32768, rdgs[160002]*range/32768);

  // reset digitizer following the transfer
  errStatus = hpe1563_reset(vi);

  // close the device sessions
  hpe1563_close(vi);// HP VXIplug&play session
  viClose(vi);// VISA session
}
```

```
//***************************************************************************

// error checking routine

void check (ViSession vi, ViStatus errStatus)
{
   ViInt32 inst_err;
   ViChar err_message[256];

   if(VI_SUCCESS > errStatus)
   {
      if(hpe1563_INSTR_ERROR_DETECTED == errStatus)
      {
         // query instrument error
         hpe1563_dcl(vi);// send a device clear
         hpe1563_error_query(vi, &inst_err, err_message);

         // display the error
         printf("Instrument Error : %ld, %s\n", inst_err, err_message);
      }

      else

      {
         // get driver error message
         hpe1563_error_message(vi, errStatus, err_message);

         // display the error
         printf("HP E1563 Driver Error : %ld, %s\n", errStatus, err_message);
      }

      hpe1563_reset(vi);// reset the digitizer
      hpe1563_close(vi);// close the digitizer handle

      exit(1);
   }

   return;

}
```

HP E8491A Specifications

Key specifications of the HP E8491A IEEE 1394 to VXI Interconnect are given in Table 4.

Table 4. HP E8491A Specifications

Product Specifications		VXI Characteristics	
Operating System	Windows 95 Windows NT	**VXI Device Type**	Message-based commander
Controllers	PC based	**Size**	C
I/O Library	SICL / VISA	**Slots**	1
Backplane	PCI	**Connectors**	P1 / P2
Maximum I/O Speed (sustained)	16-bit: 1.76 MBytes/s to PC 2.50 MBytes/s to E8491A 32 bit: 1.0 MByte/s to PC 1.0 MByte/s to E8491A	**Shared Memory**	128 kBytes
Languages	C/C++, Visual Basic, HP VEE, LabView	**VXI buses**	TTL ECL
CLK 10 Ports	Input: TTL Output: TTL		
Trigger Ports	TTL, CMOS, ECL, ± 33V		
Maximum Cable Length	72 m per system 4.5 m between devices		

Summary

The IEEE 1394 Serial Bus is versatile and affordable I/O solution for high-speed data acquisition applications involving large amounts of data. When evaluating the IEEE 1394 bus as part of your system, it is important to again consider the following:

1. Hewlett-Packard's implementation of the IEEE 1394 Serial Bus as a VXI I/O interface consists of an optional 1394-to-PCI host adapter installed in the PC, and the HP E8491A interconnect installed in the VXI mainframe. (IEEE 1394 will be a standard port on future generation PCs.) The interface is supported on Windows 95 and Windows NT platforms.

2. A VXI mainframe anywhere in the configuration can be powered on/off without affecting other mainframes. The IEEE 1394 bus automatically reconfigures itself any time a mainframe is added to, or removed from the system.

3. Data is transferred across the IEEE 1394 bus in packets. The maximum packet size is 1 kByte for 200 Mbit host adapters and 2 kBytes for 400 Mbit adapters. Transfer speed is increased during packet transfers as the protocol overhead is associated with each packet, and not with each byte transferred.

4. To take maximum advantage of the IEEE 1394 protocol, data should be transferred from VXI instruments to the PC using block transfers.

5. The IEEE 1394 bus does not provide frame-to-frame communication in multi-frame VXI systems.Thus, the multimeter and multiplexers in a VXI scanning multimeter for example, must be installed in the same mainframe. Devices sharing the VXI Local bus must also be installed in the same mainframe.

6. Register-based instruments are programmed over the IEEE 1394 interface using the instruments' VXIplug&play drivers or using register peeks and pokes.

7. Applications developed using HP VISA, HP SICL, HP VEE, and LabVIEW (running on HP VISA) are easily ported to the IEEE 1394 interface.

8. The HP E8491A has 128 kBytes of shared RAM and provides complete VXI resource manager and slot 0 capability via software that is part of the HP I_O Libraries.