TEST & MEASUREMENT WORLD
THE MAGAZINE FOR QUALITY IN ELECTRONICS

PC-BASED TEST

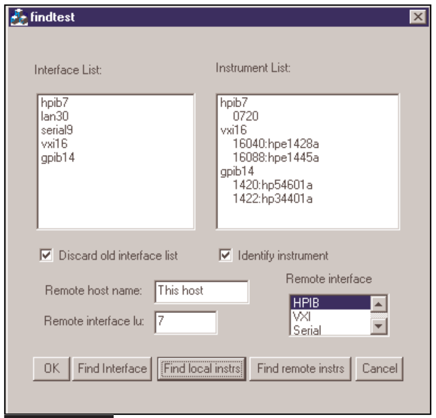
Write Your Own Instrument Finder

Tony Yokoyama
Hewlett-Packard, Loveland, CO

When writing instrument-control software, you often want to include code that finds a system's I/O interfaces and instrument bus addresses at run time. Rather than write your own instrument finder, you can use one that Hewlett-Packard makes available on the Web. Instfinder, a function library, is available for Windows as a DLL (instfinder.dll). You can download the DLL as part of a free evaluation copy of HP VEE. (See "How to Obtain the Code," p. 18.)

You can incorporate the application programming interface (API) that this library provides into your I/O applications, simplifying the discovery and addressing of instruments. Although HP developed instfinder for HP VEE, you can use the library with C or C++. I'll illustrate how to use instfinder's application programming interface (API) with Microsoft Visual C++.

The instfinder library provides functions that find instrument interfaces—IEEE 488 ports, serial ports, VXIbus, LAN—and the instruments connected to them. In addition, the library can gather information about each instrument, whether the instrument is connected to a local computer or to a remote computer. The library is compatible with HP I/O libraries and with National Instruments' NI488.2M I/O software.



```
hpib7
lan30
serial9
vxi16
gpib14
```

```
hpib7
   0720
vxi16
   16040:hpe1428a
   16088:hpe1445a
gpib14
   1420:hp54601a
   1422:hp34401a
```

[✓] Discard old interface list
[✓] Identify instrument

Remote host name: This host
Remote interface lu: 7
Remote interface: [✓ HPIB] VXI Serial

FIGURE 1. An application called findtest calls instfinder.dll, which locates and identifies instrument interfaces and the instruments connected to them.

Find Interfaces

Figure 1 shows a system's instrument configuration using an executable file called findtest, an application that provides a user interface to the instfinder library. (Findtest.exe is included as part of a download from the *T&MW* Web site; see "How to Obtain the Code," p. 18.) In Figure 1, the instrument finder discovered five interfaces: hpib7, lan30, serial9, vxi16, and gpib14. It also found instruments such as an HP34401A at bus address 22 of gpib14. The finder also identified VXIbus instruments. The instrument at bus address 16040 is an HP E1428, where the last three digits of the bus address indicate the module's VXIbus logical address.

The instfinder library follows HP I/O Library and HP VEE conventions to designate interfaces. Each type of interface has a unique address called a logical unit (LU, shown as lu in Figure 1). Commonly assigned LUs are hpib7, vxi16, and gpib14 (hpib for HP IEEE 488 controllers, gpib for National Instru-

Listing 1.

```
INTFTYPE *p, *q;
int nIntf, status;
status = 0;
p = (INTFTYPE *) findAllInterfaces(0, &nIntf, TRUE, &status);
if (p) { // success
   q = p; // save the original pointer
   for (i=0; i<Intf; ++i) { // traverse the list
      printf("Bus type at select code %d is %d\n", p->selCode, p->busType);
      // add this interface to the list of interfaces
      // find all devices at a particular interface, for explanation see
findAllInstr below
      if (p->busType == IO_NATIONAL) {
         INSTRTYPE *x, *y;
         int intfN, devs, stat;
         int flag = GETINSTRINFO | THISINTERFACEONLY;
         intfN = p->selCode;
         stat = 0;
         x= y = (INSTRTYPE *) findAllInstr(0, &nIntf, &devs, flag, &stat);
         if (x) { // success
         // handle instruments attached to NI GPIB card
         }
      }
      // done with the interface
      ++p; // go on to next interface
   } // end for
freeHeapMem((void *)q);
 //I'm done with this chunk of memory, free it
}
```

ments IEEE 488 controllers).

When you use HP I/O libraries, you can configure interfaces using the I/O Config utility, which assigns an LU to each interface. The I/O libraries use special mapping to support National Instruments IEEE 488 controllers—gpib0 is mapped to LU 14, gpib1 to LU 15, gpib2 to LU 17, and gpib3 to LU 18. If you have both Hewlett-Packard and National Instruments interfaces on your system, don't configure any interface at LU 14, 15, 17, or 18 with the HP I/O Config utility. When you use NI-488 I/O libraries, instfinder.dll will still return addresses 14, 15, 17, or 18 rather than gpib0, gpib1, gpib2, or gpib3, respectively.

The code in **Listing 1** shows how to use instfinder in an applications program. Instfinder contains two important functions: findAllInterfaces and findAllInstr. The findAllInterfaces function finds all instrument interfaces in a system, creates a list of the interface LUs, and stores that information in a data structure called INTFTYPE. The data structure contains names for the various interfaces, and **Table 1** lists those interface types and their definitions.

Table 1. Interface types supported by instfinder DLL.

IO_NATIONAL	National Instruments IEEE 488 controller
IO_82335	HP's memory-mapped IEEE 488 controller
IO_THUNDER	HP's I/O-mapped IEEE 488 controller
IO_EMBEDDED	VXIbus Slot-0 embedded controller
IO_FLEXI	VXIbus Slot-0 controller connected to a PC through IEEE 488 or MXIbus

If the findAllInterfaces function finds interfaces, it returns a pointer. The number of interfaces discovered is returned in the second parameter, which is an address to a local integer variable (&nIntf). Because findAllInterfaces creates internal data structures for subsequent instrument access, your application must call it before it calls any other function. If findAllInterfaces returns NULL, then it found no interfaces and the program in Listing 1 won't call any other function.

Make the Call

Listing 1 gives an example of how to call the findAllInterfaces function and what to do with the information it gathers. In this example, I've written a for loop that scans the interfaces for instruments. I've also included an if statement for a National Instruments IEEE 488 controller. If I didn't know which interfaces were in the test system, I'd add more for loops or use a case structure to find and identify instruments on each interface. If you download the example code, you'll get examples of how to use the case structure with several interfaces.

Listing 1 also shows the use of the findAllInstr function, which finds each instrument on an interface. Here, the findAllInstr function finds the instruments and stores their bus addresses in the INSTRTYPE data structure.

VEE's instfinder library also contains the findInstrOnThisIntf function, which is similar to findAllInstr, but it lets you find instruments on a specific interface only. You specify the interface through one of the parameters you pass to the function. In addition, the library lets you communicate with a test-system PC running the HP I/O library LAN server remotely over a LAN or the Internet.

So far, I've shown you how to find instrument interfaces and the instruments they support, but your application still needs to know what types of instruments are in the test system. You can communicate and identify IEEE 488.2-compliant instruments with the queryInstrument function. That function sends the *IDN? command to each instrument. All IEEE 488.2-compliant instruments will respond with information about the instrument. That information includes manufacturer, instrument type, model number, and more. You can parse an instrument's response to *IDN? to extract specific information about an instrument. The instrument at hpib7, address 20 in Figure 1 isn't IEEE 488.2 compliant.

I've given you a brief description of how the functions in instfinder work. For more detail, download the sample code and the library. When you download the code examples from *T&MW*'s Web site, you'll get a zipped file that contains complete C++ source code examples and detailed instructions on how they work. *T&MW*

How to Obtain the Code

To obtain instfinder.dll for Windows, download the HP VEE evaluation kit from HP's Web site, *www.hp.com/go/hpvee*. (The HP-UX version of the function library, instfinder.SI, is available only as part of the commercial HP VEE package.) To obtain the sample C++ code used in this article, visit *www.tmworld.com* and click on Software Files. Download the file associated with this article. The file contains several C++ code modules, header files, and the findtest.exe file. You'll also get a Microsoft Word document that explains in detail how the functions and code samples work and how to use them.

Tony Yokoyama *is a software engineer at Hewlett-Packard's Measurement Systems Division, Loveland, CO. His e-mail address is tonyy@lvld.hp.com.*