NCS vs. ONC

Establishing The RPC Mechanism

By Brad Harrison

Slated for distribution with HP-UX by year-end, Apollo's Network Computing System (NCS) is coming on strong. With more than 180 licensed vendors, and NCS-based applications swarming onto the market, the nearly three-year-old system is finally developing the momentum required to achieve the status of an industry standard.

But a battle is imminent. Sun Microsystems, Apollo's long-time arch rival in the workstation world, says it has the standard — not Apollo. The Open Network Computing (ONC) services available from Sun — which include its wildly successful Network File System (NFS) — offer the same functionality, according to Sun, and have enjoyed far greater commercial acceptance.

Both systems are based on the concept of a Remote Procedure Call (RPC), by which an application or procedure running on one system executes a procedure on another system. This concept is the logical final link to making the "network is the computer" model become real: programs, whether they're written by applications or systems programmers, execute across the network in the same way as they would on a stand-alone computer, but offer the programmer direct access to the rich variety of resources available on the network.

"Apollo had the chance to adopt an industry-standard RPC when it was implementing NFS on its systems," said Lou Vompo, a product marketing specialist at Sun. "Unfortunately, they chose to develop their own. Now we have two RPCs out there."

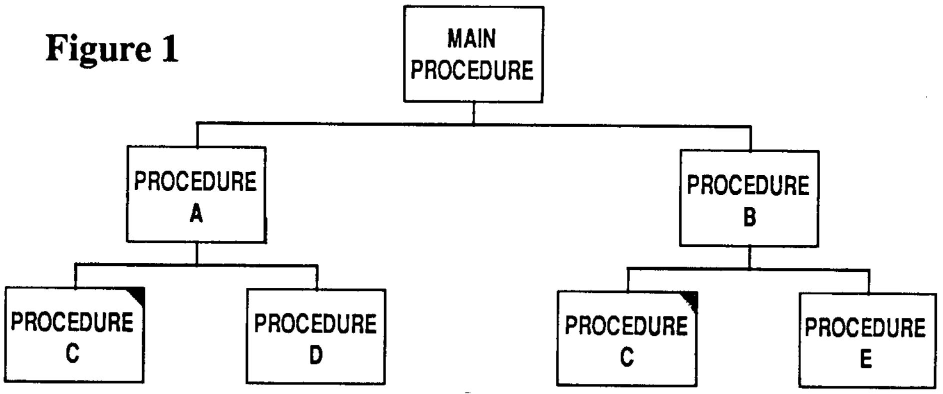
On the other hand, "Sun developed a primitive RPC mechanism when it was developing NFS," said Nathaniel Mishkin, a chief architect of NCS at Apollo. "Sun has since tried to generalize that mechanism to handle a variety of network services. They were under pressure to announce something after we announced NCS in February of 1987." Sun announced the competing ONC services in June of 1987, though NFS had been available for licensing since late 1984.

The White Paper

Apollo's Mishkin authored a paper titled "Apollo NCA and Sun ONC: A Comparison" in February of this year. Apollo has made the paper available to the industry-at-large, and it has become known as "The White Paper." In it, Apollo delineates the technical differences between NCA and ONC, and concludes that ONC offers far less performance and functionality, is more difficult to work with, and is tied to its roots in the TCP/IP world rather than offering the portability of NCS.

"The White Paper is pure opinion," said Sun's Vompo. "And nearly all of the technical criticisms are based on an old version of ONC. It's really kind of shocking that Apollo is sending this thing out to everybody — especially the press."

Figure 1



A program consists of a variety of modules — or procedures — that, under NCS, are executed on separate machines. Programs are thus distributed across the network in such a way that the various procedures make maximum use of existing resources. Procedures may be "shared" by other procedures — a shared procedure is indicated in the figure by a triangle in the upper right-hand corner.

Stalking the VAX

Regardless of their differences in opinion, Sun and Apollo have succeeded in their common goal of making the network the computer. Their common roots in the workstation world, where co-existence with other makes and types of computing equipment is a fact of life, drove them both in the same direction. Because of the capabilities they have brought to market, the need for a complicated software conversion is no longer required to add a machine to a network or to upgrade to a more powerful machine. Users may now simply "plug in" new systems and utilize their processor cycles immediately.

The idea, however, is not new. DEC has been the champion of this approach with its VAX/VMS platform, upon which any application will run whether the system be a VAXstation or a large multi-user VAX system. "Migration" in the VAX environment is a simple matter of connecting a bigger VAX to the network, and moving existing VMS applications to it. VAXes can even be "clustered" under VMS, which means that they may share common disk space, booting the same copy of the operating system and sharing files down to the record level from a large, centralized storage array.

At DEXPO in 1986, DEC announced Local Area VAXclusters (LAVcs) to tie its MicroVAXes together. Sun and Apollo had both been working on their similar technology to advance their workstation product lines, but DEC was the first to offer such a full-featured, complete package. LAVcs provide common VMS boot nodes, file sharing across the network from any node's local disk, and complete application portability. Missing was the ability to mix in machines from other vendors, as well as true processor sharing among applications.

With NCS and ONC, Apollo and Sun have now both caught up. Featuring multivendor support and a sophisticated processor sharing capability, ONC and NCS are leaving LAVcs behind.

Evolution in Network Computing

Network computing has thus seen a dramatic change over the past several years. Until recently, it has involved three basic functions: file transfer, terminal emulation, and mail. None of these functions has required truly high-performance networks. Simple file transfer merely dumps a file across the wire, hopefully fairly quickly. Terminal emulation simply needs to keep up with the user, who responds in fractions of a second rather than in the milliseconds or even microseconds at which network packets are transferred. And mail delivery is of course at the system's convenience, which doesn't present any problems since the mail is usually posted within seconds regardless of how burdened a network might be.

But now, as the network does in fact become the computer, many functions which used to occur within the computer are occurring between nodes across the network. Most importantly, RPCs are replacing local function calls within a program, and system functions like memory paging and swapping for diskless workstations are being provided by server nodes. These functions must occur within microseconds. And, of course, the ability for transparent file sharing (where any file at any node is accessible by any other node) is now commonplace.

Apollo is at the heart of this evolution in network technology, and DEC, it appears from recent announcements, will side with Apollo and NCS in the coming NCS vs. ONC battle. Who will win is surely open to debate, but it does appear that at this point that NCS has the better technology, whereas ONC has more installed product and thus a larger base to build on.

However, according to Dave Morse, a networks marketing manager at HP, co-existence is an equally feasible alternative. "Of course," said Morse, "NCS is the way to go, but that won't affect NFS. No one's trying to kill NFS." HP, according to Vompo at Sun, is the largest distributor of NFS in the world. "I don't think HP is necessarily trying to decide between the two," he said.

Indeed, Morse seemed to favor co-existence. "Our customers are combining standards on their networks. NCS complements NFS. We'd prefer if Sun would see it that way, too."

Nitty Gritty

The remainder of this article provides a technical description of NCS, noting where it — at least for now — differs from ONC. All information was derived from company literature, The White Paper, and interviews with technical personnel at Sun and Apollo.

In a large network of workstations, there will nearly always be a substantial amount of idle processor power available. The problem is how to collectively harvest it. The key force driving development of NCS was the recognition that the engineer's productivity would be greatly improved if he were able to utilize these wasted cycles to make his applications run faster.

NCS was originally conceived of as a system to automatically partition a program and distribute it across the network. It would thus provide the user with optimized application performance, given current network utilization. Complete realization of this goal is still several years away, but already NCS is being used in many installations to increase performance and functionality for certain classes of applications.

For example, at Rohm and Haas Company of Philadelphia, a manufacturer of speciality chemicals, a program used by chemists to display and rotate molecules was ported to a 1-MIPS Apollo workstation from an IBM MVS-based mainframe, where it had been developed. Shockingly, on the workstation the program required more than an hour of CPU time to draw any molecule composed of more than 50 atoms. This was, of course, unacceptable.

Using NCS, programmers re-partitioned the FORTRAN program so that the bulk of the program's mathematical operations were executed as subroutines on a network minisupercomputer. Chemists now interactively display and manipulate chemical structures with no delay.

Another example application is a realtime data acquisition product called RTNode from Obsidian Computers in Cambridge, Mass. RTNode is implemented as a network of workstations collecting data over the network from a variety of data acquisition devices. Under NCS, the workstations run common procedures on embedded microprocessors in the data acquisition devices.

Apollo is currently at work on extensions to NCS that will make it more appealing to the financial marketplace. These enhancements include outfitting the system with database development and implementation tools, and support for the languages normally found in business environments.

Installing NCS

NCS has been licensed by Apollo to companies including DEC, IBM, Sun and Cray. A license allows the vendor to market

(See NCS, next page)

Figure 2

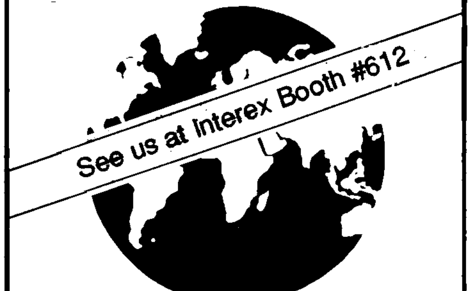
```
[uuid (334033030000.0d.000.00.87.84.00.00.00),
  version (1)]                                              (1)
interface bank {
    import
            "nbase.idl";                                    (2)
    typedef                                                 (3)
                                      long int    bank $acct_t;
    typedef
            char bank$acct_name_t[32];
    void bank$deposit (                                     (4)
            [in]        handle_t            h,
            [in]        bank$acct_t         acct,
            [in]        long int            amount,
            [out]       status_$t           *status
    );
);
```

The interfaces between NCS clients and servers are described using C-like code that is compiled into C code by the NIDL compiler. This NIDL code specifies an interface called bank that calls a remotable procedure called $deposit.

BUY • SELL • MAINTENANCE

HP1000
HP3000
HP9000
HP Compatibles

World Class Service



World Class Savings

- SYSTEMS
- DISC DRIVES
- TERMINALS
- MEMORY
- I/O EXPANSION
- PRINTERS
- PLOTTERS

LARGE INVENTORY
GUARANTEED MAINTENANCE
BEST PRICES

COMPUTECH
SYSTEMS CORPORATION
2721 152nd Ave. N.E.
Redmond, WA 98052
(206) 883-4107
1-800-882-0201 (US)
FAX: (206) 881-2482

Response No. 16

DSP-300
Signal Processing Coprocessor for Hewlett-Packard Series 200/300 Computers

HP-UX Driver Now Available!

Easy to Use • Simple, high level commands from BASIC or Pascal.

High Precision • 144 to 336 dB dynamic range. Processes real or integer arrays.

High Speed • Performs 1024 point complex FFT in 3.39 msec.

Programmable • General purpose DSP chip: Motorola's DSP56001.

Large Arrays • Up to 64K of 24 bit data memory (16K point FFT's).

High Throughput • Dedicated 24 bit parallel port for high speed signal I/O.

Large Program Memory • Up to 64K program memory.

Flexible • Supports DMA and interrupts.

Compact • Occupies a single DIO slot.

Open Architecture • Custom applications supported with complete documentation.

Multilingual • BASIC and Pascal interfaces supplied with source code.

Available • Now shipping from stock.

Coming Soon • Real-time 14 bit analog I/O. Ask about the DSP-300-A14.

DSP-300 brings high precision, high throughput, signal processing power to the Hewlett-Packard Series 200/300 engineering workstation. Combining Motorola's state-of-the-art DSP56001 signal processing CPU with up to 128K words of zero wait-state memory, DSP-300 greatly accelerates compute-intensive tasks commonly encountered in signal processing systems. Ariel-supplied software permits complex signal processing functions, such as the FFT, to be called with a single line of BASIC or Pascal code.

Ariel Corporation is dedicated to providing you with the best values in high performance DSP products. Our products are designed, built and maintained in the U.S. The best support in the industry is always at hand. Ariel's products are sold directly throughout North America, and are available worldwide, through our international dealer network.

Ariel Corporation
433 River Road
Highland Park, NJ 08904
Telephone: 201-249-2900
Fax: 201-249-2123
Telex: 4997279 ARIEL
DSP BBS: 201-249-2124

Ariel

Response No. 17

NCS

(Continued from previous page)

the NCS Network Interface Definition (NIDL) compiler, the run-time code, and a distributed application called the location broker. All components are written in C, so are easy to port to new systems. NCS is protocol independent, though Apollo recommends that it be implemented directly on top of a simple datagram protocol like Internet Protocol (IP), which simply provides the network-level services of addressing, routing, and packet fragmentation and reassembly. To use a higher-level protocol like Transmission Control Protocol (TCP) or DECnet imposes additional overhead that impacts performance.

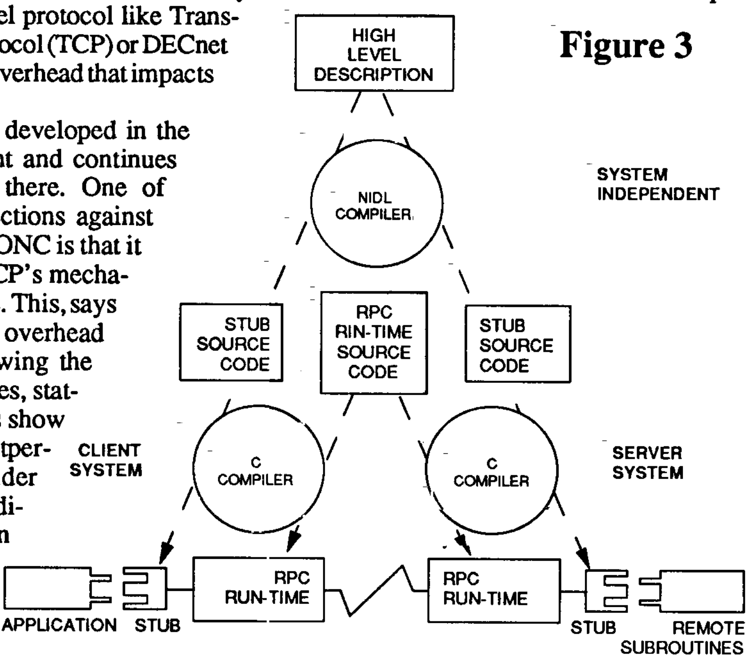
Sun's ONC was developed in the TCP/IP environment and continues to show its roots there. One of Apollo's main objections against the functionality of ONC is that it relies on some of TCP's mechanisms to be effective. This, says Apollo, adds to the overhead of ONC RPCs, slowing the system. Sun disagrees, stating that benchmarks show ONC under TCP outperforming NCS under UDP (which is a radically stripped down version of TCP, offering little more than IP, which NCS requires to run).

In designing NCS, Apollo was very careful to ensure maximum performance in all respects. NCS eliminates the overhead of any operation that, in certain circumstances, imposes an additional burden on the network for no good reason. For example, NCS distinguishes between calls to *idempotent* and *non-idempotent* RPCs. An *idempotent* procedure is one that may be executed more than once without side effects, such as simple mathematical calculations; on the other hand, a *non-idempotent* procedure has significant consequences, such as to the data in a file update operation. NCS ensures that the calling program carefully tracks the progress of *non-idempotent* operations, whereas this overhead is eliminated if the program has made a call to an *idempotent* procedure. By handling special cases such as these, Apollo has built maximum performance into NCS without burdening the programmer.

In ONC, if *non-idempotent* procedure calls are to be carefully tracked, the protocol used must be TCP. If ONC RPCs are implemented at the datagram level, no mechanisms exist to ensure that they are handled appropriately.

The NCS run-time environment implements the basic RPC mechanism. It interfaces to and supports the network protocols in use and passes and catches datagrams. It handles data representation and conversion, whereby data received/sent from/by a client or server is reformatted to match the computer's native data types. It opens and closes communications sockets (similar to files). The run-time environment shields the programmer from the details involved in enabling his programs to make RPCs, and causes procedures to appear as though they were being called locally.

Figure 3



NCS is structured to be system-independent and easy to use. Implementing stubs, the designers masked the differences between procedures executing remotely and those executing locally.

A key performance issue for the run-time environment is how it goes about converting data to native machine data formats. Sun and Apollo differ sharply here. Sun, with its external data representation (XDR) model, has dealt with the problem by what's known as the canonical approach: data is converted to a common format for transmission across the network, then reconverted by the target machine into its native data types. This process works especially well in a multi-vendor environment because all nodes act on the data in a common way.

Apollo disagrees with this approach because conversion is required even if the two machines have the same architectures, and has opted instead to send data in the format

(See NCS, page 36)

GKS

(Continued from page 29)

For example, the PICK function may work with either a keyboard or a mouse attached to an HP 9000. GKS maintains a list of the physical input devices available with each workstation in an application environment. An application can be easily modified to handle changes in input hardware.

Conclusion

The facilities available in GKS provide a graphics programmer with a complete and rich development environment. It allows application programmers to concentrate on developing specific applications and take advantage of the underlying device support available through GKS. This approach to developing graphics applications can result in reduced program development time and lower long-term support and maintenance costs.

Gary Shelef is founder of the Workstation Consulting Group, an Ann Arbor, Mich. firm specializing in graphics programming, market research and technical writing for the UNIX workstation market.

Dr. Sahib Dudani is president of Advanced Technology Center (ATC), a maker of graphics software systems for HP, Apollo, and other platforms.

Learning More about GKS: Readers interested in learning more about GKS can consult one of following references or ATC.

Enderle, G., K. Kansy and G. Pfaff, *Computer Graphics Programming (GKS -The Graphics Standard)*, Springer-Verlag, New York, 1984.

Salmon, R. and M. Slater, *Computer Graphics - Systems & Concepts*, Addison-Wesley, Workingham, England, 1987.

Hopgood, F.R.A., D.A. Duce, J.R. Gallop and D.C. Sutcliffe, *Introduction to the Graphical Kernel System (GKS)*, Academic Press, London, 1986

Foley, J.D. and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, Mass., 1982.

Brown, M., *Understanding PHIGS*, Megatek Corp., San Diego, Calif., 1985.

Arnold, D.B. and P.R. Bono, *CGM and CGI*, Springer-Verlag, New York, 1988.

UPGRADE!

IBM PC to HEWLETT-PACKARD File Copy

- *Up to 10 times faster!*
- *Easy to use!*
- *Always dependable!*
- *Call for free demo!*

IBM PC to HP FILE COPY $495

OSWEGO SOFTWARE
P.O. Box 310
Oswego, IL 60543, U.S.A.

Specialists in disk file copying and program translation for HP technical computers.

Call us.
312/554-3567
FAX 312/554-3573

Response No. 26

File Management Library

MODULE LIBRARIAN — 2.0 —

BASIC Windows — 3.0 Human Interface Library

BASIC ESSENTIALS - 3.0

BASIC Windows 3.0

Imagine...

Window style menus, forms-based entry, text, and icons under BASIC 3.0-5.1 on HP200/300 Series computers.

Generate professional user interfaces with simple CALL's in minutes... not months!

Built-in support for most options including ALL monitors, touch screen, knob, mouse, and VIPER card... at the right price!

The Complete Human Interface.

!! DEMO Now Available !!

Module Librarian 2.0
Program development, management and maintenance facility.

File Management Library
Self-documenting ASCII & BDAT file handling routines.

BASIC ESSENTIALS 3.0
Money saving bundle: BASIC Windows 3.0, Module Librarian, and the File Management Library. **Call Now!**

MR
MODULAR RESOURCES CORPORATION
P.O. Box 718 * Lowell, MA 01853
Or call us at: (617) 348-2604

Response No. 25

NCS

(Continued from page 24)

of the source machine with the data thus identified; it's then up to the target to reformat the data stream according to the formats of the data types it uses.

But this requires the target to have the capability of converting many different native formats into its native data types. If there are many different types of machines on the network, the ability to make the conversion could require substantial system resources. Yet Apollo claims that the overhead of two conversions — especially when no conversions are necessary — is worse. This is probably the most important — and stickiest — areas of the NCS vs. ONC debate.

The location broker is a distributed application that tracks where all procedures are located on the network. When a client program makes an RPC, it actually invokes the location broker, which then returns the address of the server most suitable to execute the call. Under NCS, clients make no references to specific machines. It is thus object-oriented in its binding method, as opposed to forcing the programmer to become involved with physical addresses on the network (though he may in fact specify a given machine, as described below). Servers "list" their procedures with the location broker, and keep the broker updated on available resources.

Sun's counterpart to the location broker in ONC is known as Yellow Pages. In The White Paper, Apollo criticizes ONC for not having the same dynamic functionality as the location broker, that Yellow Pages needs frequent tinkering by a network administrator to keep up with activity on the network. In addition, Apollo claims that ONC doesn't really treat RPCs as objects, but forces the programmer to be more concerned with actual locations of network resources. Sun claims it has recently made significant improvements to Yellow Pages so that it now is capable of the same functionality and performance as the location broker, and that its binding methods are now as object-oriented as NCS's.

(See NCS, next page)

NCS

(Continued from previous page)

Programming Under NCS

As shown in Figure 1 (on page 22), a typical application program is composed of several modules. A main module is implemented to call the major components of the program, which may in turn call their component modules. In FORTRAN, these modules are subroutines. In C, they're functions. Pascal supports functions and procedures, the difference being that a function may return a single value where a procedure may return many values. Under NCS, all modules are referred to as procedures, and the calls that occur between modules are interfaces.

According to the tenets of structured programming, a programmer attempts to break his problem into sub-problems, each of which may be solved separately using a procedure. The interfaces between the resulting modules are the data types and data structures passed back and forth.

Under NCS, the programmer is confronted with more options for procedure partitioning than when he was running his program on a single machine. Now he must physically pass entire arrays, trees and lists of data between machines, since procedures don't have access to common memory. Whereas before he may have broken his program into modules according to logical division of program operations, he must now consider data transfer vs. compute time, and which machines on the network might be more appropriate than others for certain procedures. NCS supports some parallel synchronization tools so that RPCs can be made in parallel to increase application performance.

In many cases the programmer will not be starting from scratch, but will draw on procedure libraries that specify the procedures available across the network. It makes no difference what language these procedures were coded in, but they must be able to properly operate on the data types provided by the client in the procedure argument lists.

Using NIDL, the programmer must develop the code that specifies the interface for each of the client-server pairs he has written (see Figure 2 on page 23). In the example, (1) defines the universal unique identifier (UUID) by which this interface, bank, is known. UUIDs are fixed-length (128-bit) identifiers that are formed by combining the network address (with protocol suite number) of the system making the UUID and its the current time. Every interface has a unique UUID. When the location broker is presented with a UUID it return the address of a server that has registered with the location broker to handle operations for the specified object. The client then makes its RPC to that server.

(2) declares the interfaces upon which this interface is dependent. The import statement is similar to the #include statement in C, except that the named interface is not physically compiled with this interface. The importer may refer to the data types and constants referred to in that interface. (3) defines data types for the interface with typedef (again, as in C), and (4) defines the operation itself. Here, interface bank calls remotable procedure $deposit.

The NIDL compiler supports three types of binding: explicit binding, whereby the NIDL specification states exactly which host to use; implicit binding, whereby the location broker checks to see where the procedure is available for execution by the RPC; and automatic binding, which is used if a client program needs to access different hosts at different times using the same RPC.

Once he has appropriately coded his procedures and their NIDL interface specifications, the programmer uses a file transfer utility to move the source code to the machines where it will be used. He separately compiles the NIDL descriptions using the NIDL compiler, which produces C source code. This source code is then compiled with a C language compiler, producing the stubs that interface the client and server procedures to the NCS run-time environment. The process is illustrated in Figure 3 (page 24).

The process of program development under both NCS and ONC is complicated compared to writing programs that will run on a single machine. Sun and Apollo both claim that many improvements are on the horizon. The issue of which complier is better — NIDL or Sun's Remote Procedure Call Language (RPCL) compiler — is another area of debate, and is probably best addressed by programmers in the field using the products, though they are relatively few at this point.

But applications for both NCS and ONC are showing themselves daily. Most network functions are based on the concept of an RPC in one form or another, and as RPCs become easier to work with and networks are designed for maximum RPC performance, the majority of system and applications software will be developed using these tools. As to which system will win, the issue is far from settled. For the time being it's enough to know that some important network tools have arrived to help us be more effective in our work, and that they are finding industry-wide support.

Brad Harrison is a freelance writer based in Fort Collins, Colo. He has worked in the computer industry for five years.

Product Availability

This space is a contact reference for the third-party products listed in HP D&A news reports and features. The publisher assumes no liability for errors or omissions.

ATC	**Response No. 200**	5711 Slauson Ave, Suite 238, Culver City, CA 90230
Bering Industries	**Response No. 201**	246 East Hacienda Ave., Campbell, CA 95008
Cadam Inc.	**Response No. 202**	1935 N. Buena Vista St., Burbank, CA 91504
Cadence Design Systems	**Response No. 203**	555 River Oaks Pkwy, San Jose, Calif.95134
Compact Software Inc.	**Response No. 204**	483 McLean Blvd., Paterson, N.J. 07504
EEsof Inc.	**Response No. 205**	5795 Lindero Canyon Rd., Westlake Village, CA 91362
Electrical Engineering Soft.	**Response No. 206**	4675 Stevens Blvd., Santa Clara, CA 95051
Eyring	**Response No. 207**	1455 West 820 North, Provo, UT 84601
Gateway Design Automation	**Response No. 208**	2 Lowell Research Center Dr., Lowell, MA 01852-4995
Laboratory Technologies	**Response No. 209**	400 Research Dr., Wilmington, MA 01887
LSI Logic	**Response No. 210**	1551 McCarthyBlvd., Milpitas, CA 95035
Mentor Graphics	**Response No. 211**	15220 N.W. Greenbrier Pkwy., Ste. 300, Beaverton, OR 97006
Meta-Software	**Response No. 212**	50 Curtner Ave., Suite 16, Campbell, CA 95008
Mitsubishi Printers	**Response No. 213**	520 Madison Ave., New York, N.Y.10022
Oracle Corp.	**Response No. 214**	20 Davis Dr., Belmont, CA 94002
Pinnacle Micro	**Response No. 215**	15265 Alton Parkway, Irvine, CA 92718
PlotPro Inc	**Response No. 216**	2333-D Wirtcrest Ln., Houston, TX 77055
Seikosha	**Response No. 217**	1111 Macarthur Blvd., Mahwah, N.J, 07430
Template Graphics	**Response No. 218**	9685 Scranton Rd., Suite 150, San Diego, CA 92121
TransEra	**Response No. 219**	3707 North Canyon Rd., Provo, Utah 84604
Tymlabs	**Response No. 220**	811 Barton Springs Rd., Austin, TX 78704