

HP 9000 WHITE PAPERS

These white papers cover specific details on the HP 9000 that we feel will be useful, but are not appropriate for the sales training manual. They provide information that may be useful in particular sales situations. Each white paper is summarized on one or two pages at the start; these pages can be used as slides for a presentation on the white paper topic. For a quick overview of the white papers, just review the slide summary. The Datacomm and HP 9000 strategy papers are all strategic, covering future plans for the 9000. The other white papers cover technical details. The technical papers can be distributed to customers if you feel it appropriate; do not distribute the strategic papers to customers. These white papers contain the best available information as of September 15, 1982; contact Northern Colorado Sales Support for most current data.

CONTENTS:

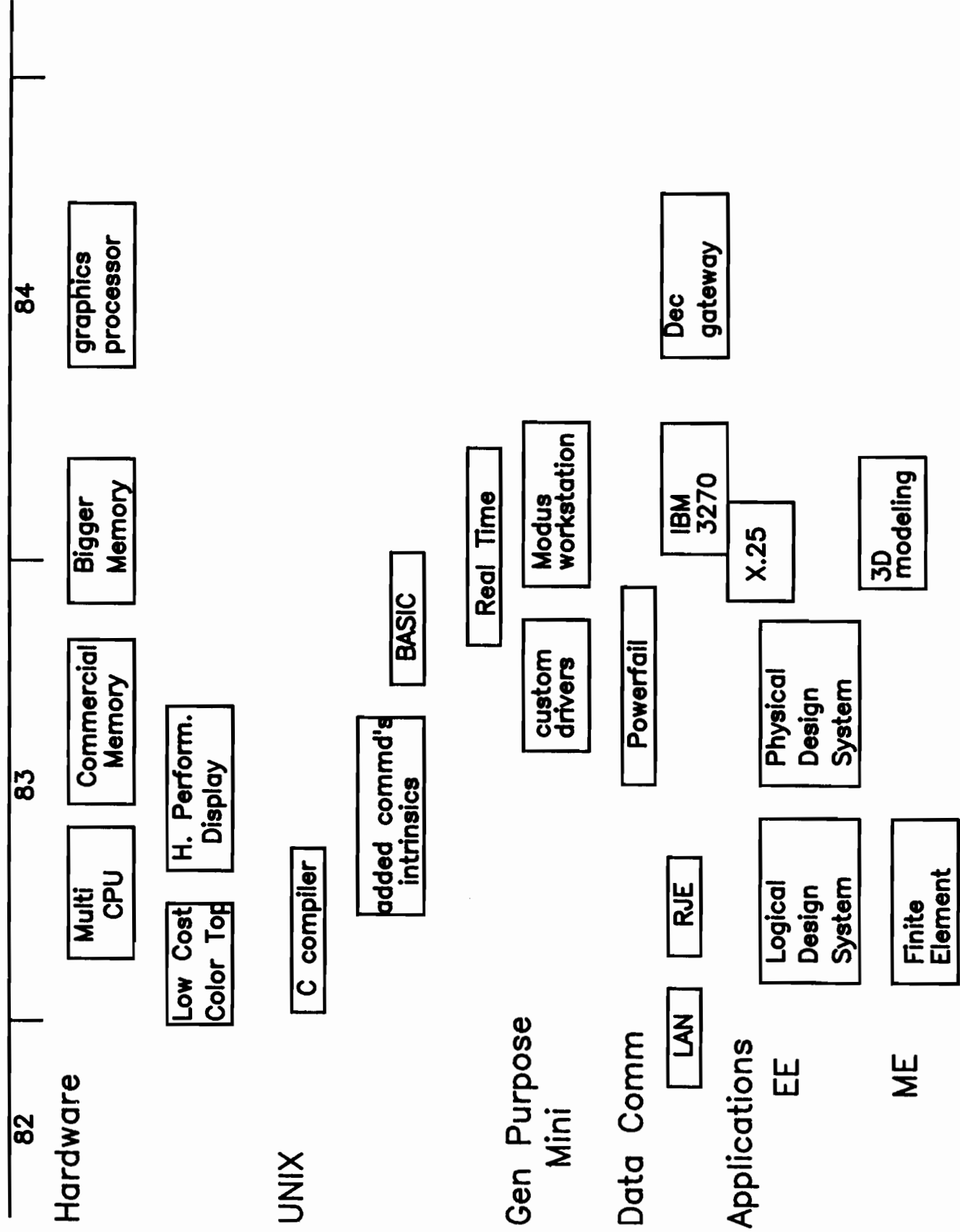
HP 9000 PRODUCT FAMILY STRATEGY
HP 9000 DATACOMM
HP 9000/HP-UX
HP 9000 APPLICATION SOFTWARE
HP 9000 GRAPHICS
HP 9000 I/O
HP 9000 AS A GENERAL PURPOSE MINI
HP 9000 vs. 9845 BASIC
HP 9000 vs 9845 BASIC I/O

HP 9000 vs 9845 BASIC GRAPHICS
HP 9000 vs 9826 BASIC
FORTRAN/9000
PASCAL/9000
IMAGE/9000
HP 9000 - HP 1000 COMPARISON GUIDE
GLOSSARY OF TERMS



HP 9000 PRODUCT FAMILY STRATEGY WHITE PAPER

Projected HP 9000 Enhancements



HP 9000 PRODUCT FAMILY STRATEGY WHITE PAPER

The following white paper describes the HP 9000 Product Family Strategy and enhancement directions. The intent is to describe each major functional element of the HP 9000 System enhancements to provide Field Management with information which would guide them in talking to Major Accounts who have a long term requirement for which they are evaluating HP's offering. This is not intended to be a handout for customers.

MULTIPLE CPU'S

The HP 9000 architecture supports multiprocessors on a common memory processor bus. This is a major breakthrough in hardware/software technology which provides for incremental performance by simply plugging in additional processor finstrates. Each CPU has a clock rate of 18 MHz while the memory processor bus performs at 36 Megabytes/second. The processors share memory but contention for the memory and bus is minimized by the memory processor bus architecture and speed. The benchmarks indicate the second CPU increases the throughput 1.8 times; the third CPU increases the throughput 2.7 times. Maximum efficiency in multiple CPU systems is realized when multitasking or multiuser requirements are present. Software written on a single CPU system will run on a MCPU system without rewrite. The Operating System understands MCPU availability and program segmentation to take advantage of these MCPU's. Customers should expect that a continuous serial 10,000 line FORTRAN program with no subprograms will not experience increased performance throughput. The functionality of this feature is only effective for multitasking or multiuser systems which are the primary requirements for increased performance.

NETWORKING

The HP 9000 networking capabilities are extensive, providing both multivendor networks and HP Proprietary networks. Ethernet is available at introduction

and offers remote file access, network file transfer and interprocess communication. The Ethernet product will be accessed by HP-IB interfaces which connect to an Ethernet converter box. This Ethernet product will evolve into an HP LAL which will be directly accessed via special HP-IO interfaces. Software will be directly transportable from the initial Ethernet product to the HP LAL. HP's intent is to follow the IEEE 802 standard for the HP LAL; we intend to use the forthcoming industry standard LAL chips for the interface. Ethernet is a 10 Megabit link with effective file transfers at 100K bits/second. The LAL will also be 10 Megabit; however, it is intended that improved software will permit better effective file transfer performance. Ethernet will be supported on both HP 9000 and HP 3000 computers but 9000's can communicate only with 9000's. HP LAL will be able to include 9800, 9000, 3000 and possibly 1000 systems on the network. The Shared Resource Manager will support the HP 9000 and will have the same features and limitations as previously introduced with the 9800. The Shared Resource Manager will be supported on the HP LAL when it is available.

DATA COMM GATEWAY

The HP 9000 will support gateways to IBM and DEC computers. Asynchronous terminal emulators are available at introduction, providing log-in terminal capability to host systems. 2780/3780 RJE gateways will be available for UNIX Operating Systems in Q3 of '83. This product will be similar to the 9845 RJE product with up to 9600 bits/second performance. A 3270 gateway will be available in '84 to provide an interactive link to IBM systems. There are no plans at present to provide SNA service from the HP 9000 to IBM networks.

The DEC gateway will be available in '84. The specific services will be remote file transfer and possibly program-to-program communication. Exact details of this link are not determined at this time. Possible implementations include VT100 emulation, Ethernet software, an X.25 link or a limited DECNET node. The full DECNET functionality would be impossible to support

DATACOMM GATEWAY (Continued)

and is not appropriate since we are trying to provide a way to interface to DEC installed systems. Public Data Communications links will be provided by X.25. This will be developed in HP Grenoble and will be available in Q4 of '83.

In general, customers want to know what our network and Datacomm strategy is and frequently will not purchase all of the functionality at the time they purchase their first system. The Datacomm Gateways have been prioritized in accordance with customer inputs and the engineering challenge associated with each.

UNIX ENHANCEMENTS

9000 HP-UX Operating System is based upon Bell Labs/Western Electric UNIX System III. This provides a basis for importation of software written in a standard language and UNIX System III (i.e., System III UNIX transported to the HP 9000). The C Language is one of the cornerstones of the UNIX System. The HP 9000 will offer a C compiler. This compiler will permit easy transport of standard UNIX software to the HP 9000. When appropriate HP-UX system documentation is added (later in '83), C will provide the advanced UNIX user with access to the native Operating System in order to be able to extend the functionality of UNIX, write special drivers, and import hardware-dependent programs that have been written in C on other UNIX systems. Other UNIX enhancements include additional commands and intrinsics which will be supported in Phase Release 2.0. A detailed list of which commands will be included in that group is available from your local SE or ESO Sales Development. In addition, UNIX will be enhanced to provide a Real Time Functionality. The first step is to add capabilities similar to those offered by the BASIC Language System. It is not intended to replace the RTE Functionality on the HP 1000. This topic will be the subject of another white paper which will be available from DSD to describe the RTE migration and emulation plans for the HP 9000. In Q4 of '83, BASIC Language compatible with the HP 9000 BASIC System will be offered on HP-UX.

GENERAL PURPOSE OEM COMPUTER

The HP 9000 Family will be enhanced over time to further increase its appeal and functionality for the general purpose OEM marketplace. The UNIX Operating System is one of the strengths of the HP 9000 for the OEM and general purpose user. The broad functionality, programmer productivity and multivendor transportability potential are important criteria for choosing the HP 9000 and UNIX. The standard languages — FORTRAN, PASCAL and C — offer additional enticement. Real Time Functionality has also been described in the previous UNIX Enhancement topical area but is worthy of mention here as an important aspect of the general purpose OEM computer requirements. Access to the backplane is an area of considerable interest for selected OEM's, particularly where special drivers have been written for such things as flatbed plotters, specialized workstations or specialized Data Acquisition and control systems. This will be offered Q4 of '83 through both PASCAL and C. An assembler — Assembly Language programming capability — will be provided in a Modus Workstation Q1 of '84. Modus is the software development workstation which was used for the development of the HP 9000 Operating System software, based on MODCAL, "MODULAR PASCAL," and some special software engineering tools to access the Kernal Operating System and to write Assembly Language programs that extend the function of the HP 9000 and/or to tune specific applications programs to optimize them for performance-time critical operations. Power Fail and Auto Restart capability will also be offered late in '83. The availability of Power Fail/Auto Restart is contingent upon commercial RAM being utilized in the HP 9000. The plans are to make commercial RAM available; however, it's likely that it will be of a lower performance than the NMOSIII RAM but for single CPU's this should be transparent to the user. The Power Fail/Auto Restart capability is most often requested for the factory and plant automation supervisory computer application area. Approximately 50% of the HP 1000 customers buy Power Fail and Auto Restart. It's an important extension of the HP 9000 for the supervisory computer and others who may require this capability.

MEMORY EXPANSION

The 2.5 MByte Internal Read/Write Memory presents no limitations for the single user workstations but in some memory-intensive computational applications in a multiuser environment or multiple CPU environment it may prove to be a limitation. There are plans to expand the memory to 5 MBytes and beyond, consistent with the availability of higher density memories and the requirements of our customers. A 5 MByte memory option will be available in late '83 or early '84 and will fit in either the integrated 9000 workstation or the modular 9000 system. The memory expansion will be implemented through a packaging change and is not dependent upon denser RAM's available in the marketplace. Upgrades will probably be available by the time customers find that they need additional memory over the 2.5 MBytes available at introduction.

APPLICATIONS

There are a number of CAE applications under development within HP. HP Design Aids and ESO West are responsible for the CAE Electrical Engineering applications. These include the Logical Design System which consists of DRAW and SPICE, a schematic capture and circuit simulation capability, as well as a Physical Design System which includes the printed circuit board layout, packing and routing. It's an integrated system; a common Data Base will ensure that if the schematic is correct, the physical printed circuit design will be correct. This will be a major contribution to the PC design process. It will provide interfaces to the Manufacturing quadrant for automatic drilling of printed circuit boards and automatic testing of printed circuit boards as well as interfaces to the upper half of the MPN quadrant including materials management. This capability will be available in two phases — the Logical Design System in the second quarter of '83 and the Physical Design System in the third quarter of '83. HP Boeblingen is responsible for the CAE Mechanical Engineering applications. The 2-D Drafting System and Finite Element package has been introduced. A 3-D Modeling Package is under development which will be available in Q1 of '84. These applications

APPLICATIONS (Continued)

like the CAE-EE, function on a common Data Base so that data shared between the analysis, design and documentation phases can be output to the Numerical Control and Materials Management Systems. The Computer Aided Work, Computer Aided Test Applications are the responsibility of DCD and the Technical Applications Group. A series of applications including the EGS, CAP Statistics and Project Management will be available during the second half of '83 and beyond.

Applications are key to the success of the HP 9000 and although our HP Proprietary applications are significant, the base of CAE software which is available in the marketplace and could be aimed for the HP 9000 through Third Parties is enormous. We have an aggressive recruiting program in place to have the widely used and important applications software and tools, primarily written in FORTRAN and most prominently on the DEC computers, moved to the HP 9000. These include such packages as CADAM, the Computer Aided Design package originally developed by Lockheed and running on large IBM mainframes; NASTRAN, by MacNeal-Schwendler, the most comprehensive and widely used Finite Element Analysis Package; Softool 80, a software engineering workstation toolbox with significant tools to facilitate transportation from mainframes and DEC computers as well as tools to facilitate the design of new software applications on the HP 9000; IMSL, a large FORTRAN Math Library; SPSS, a FORTRAN Statistics Library; the MUSE Text and Document Processing software; DISPLAA, the graphics charts and graphs applications package from ISSCO and many others. This is just a sampling of the Third Party software which we have under negotiation. If you or your customers have specific software for which they are interested, and a decision about the HP 9000 is contingent, please contact the Sales Development Organization.

DISPLAYS

The display offering for the HP 9000 is excellent for general engineering -- including the high performance monochromatic and the high performance color graphics displays plus the availability of an OEM monitor for Computer Aided Design. In early '83, a low cost color top, similar to that planned for the 9836 will be introduced for the HP 9000 Model 20. Also planned for Q2 of '83 is an OEM Graphics System with 19 inch, 1000 x 1000 Pixel Display with 256 colors and 4000-5000 vectors/second performance. We have under development a product which will include a functional graphics processor, EDISON, which will be a remicrocoded 32-bit CPU dedicated to graphics processing and an HP display, 19 inch, 1000 x 1000, with an overall performance of 20,000 vectors per second. The target price is \$10,000 to \$15,000.





HP 9000 DATA COMM

HP 9000 SYSTEM – NETWORKING STRATEGY

OBJECTIVE

Support the HP 9000 systems in a computer aided engineering applications in the Manufacturer's Productivity Network.

- * Access to data and programs on large mainframes by series 9000 CAE workstation cluster is a necessary ingredient for success.

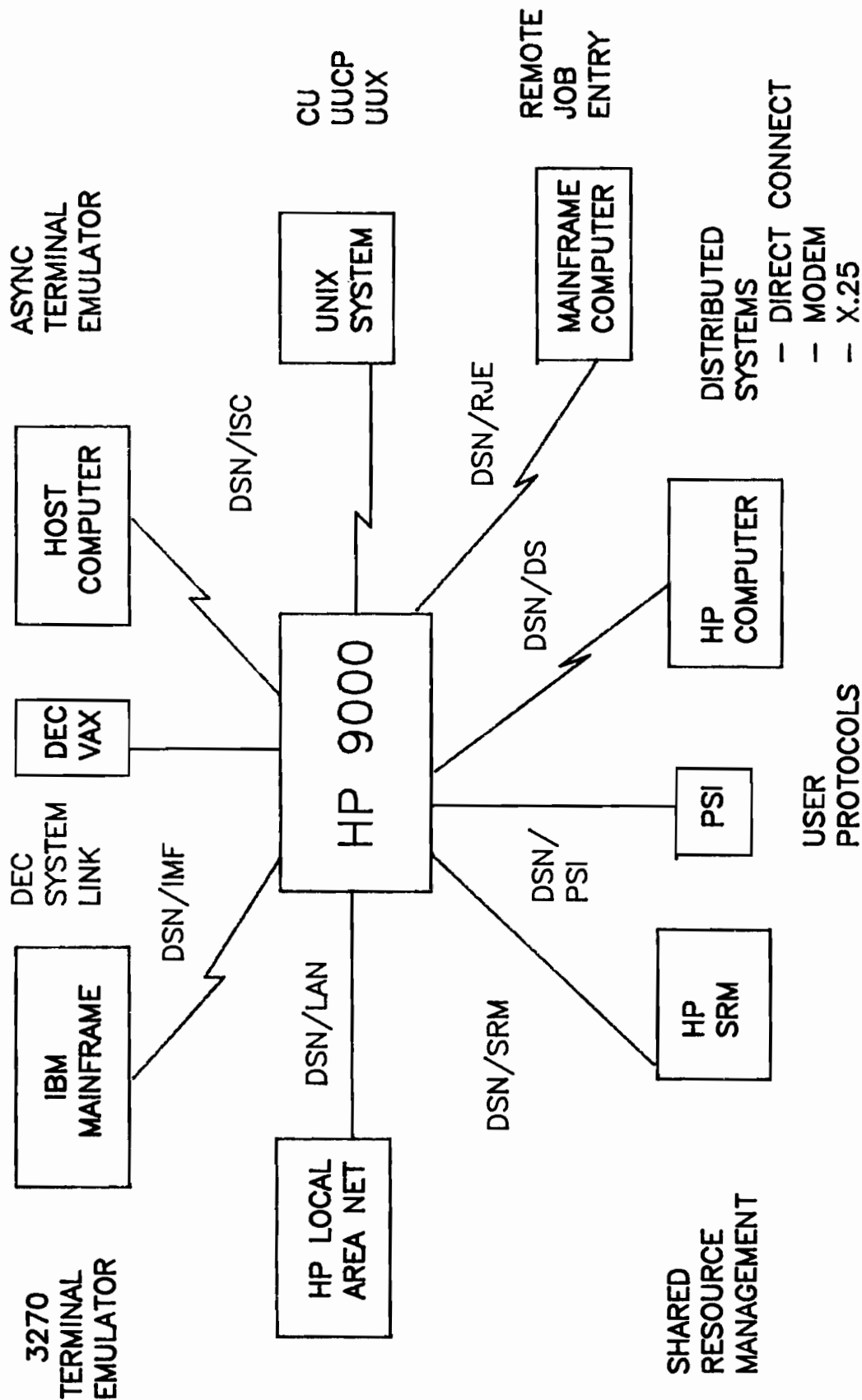
STRATEGY

1. High Performance HP Clusters – Develop high performance local networks to interconnect HP 9000 workstations in clusters: emphasize sharing disc, printer, and plotter resources, tightly coupling CAE applications, and high speed movement of data.
2. HP – Distributed Systems Network – Provide standard services and network links for interconnecting HP 9000's to HP-MPN applications.
 - * Links include remote point-to-point, Public Data Nets (X.25), and Ethernet-IEEE 802 10 Mbps baseband link.
3. NON-HP Networks – Provide connections to non-HP mainframes, specifically IBM mainframes and DEC VAX. Users can access data and programs on a central system & move results to and from HP 9000 systems cluster.

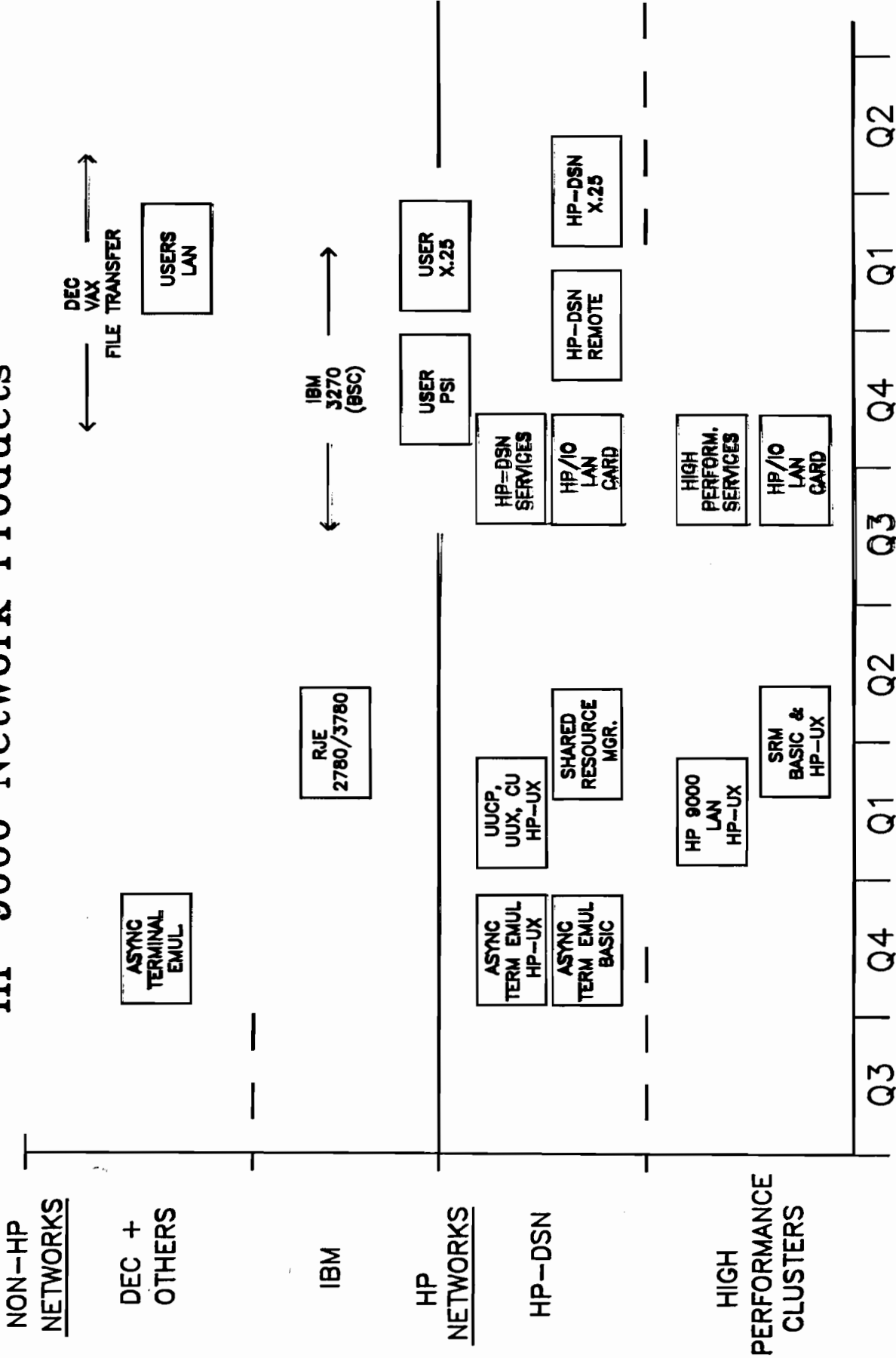
HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HP 9000 DISTRIBUTED SYSTEMS NETWORK PRODUCTS



HP 9000 Network Products



1984

1983

1982

USER - OEM, 3RD PARTY, LARGE END USER LAN- LOCAL AREA NETWORK (ETHERNET/IEEE802)

HP 9000 DATA COMMUNICATIONS/NETWORKING STRATEGY

OVERVIEW

Data communications and networking are the foundation for Hewlett-Packard's Manufacturer's Productivity Network (MPN) and an important part of the HP 9000 Computer Aided Engineering Workstation strategy. The HP 9000 Program will provide continuing enhancements to its data communications products for clusters of CAE workstations on a local area network, connections to other Hewlett-Packard mainframes and applications in MPN, and access to user data and programs on non-Hewlett-Packard mainframes. The networking strategy for the Series 9000 is being developed as part of the Hewlett-Packard Distributed System Network.

When HP 9000 systems are combined with a high performance local area network they become a competitive alternative to centralized time-sharing systems. This clustering of workstations emphasizes sharing of resources (like discs, printers, and plotters), cooperating applications programs, and high speed movement of data between workstations. A series of networking products will be provided over the next three years to achieve this concept. Other networking products ('gateways') will be developed to move the results from the tightly coupled workstations to Hewlett-Packard, IBM, and Digital Equipment mainframes.

HEWLETT-PACKARD DISTRIBUTED SYSTEM NETWORK

At introduction (Q1 1983) the HP 9000 system will have two Asynchronous Terminal Emulators, one for HP-BASIC and the other for HP-UX, a UNIX based operating system. The Terminal Emulators are the most flexible and lowest cost connections to HP and non-HP mainframes. Each emulator has a file transfer capability using ASCII input/output files, industry standard control handshakes, and a variety of configuration options. For HP-UX, the Unix-to-Unix-Copy (UUCP) and Call-Unix (CU) utilities will be supported for connecting to any standard Unix system.

In January of 1983, the HP 9000 will offer an industry standard 10 mega-bit baseband (Ethernet) network and high performance network software. The network software will include Network File Transfer and Inter-Process Communication, plus a Remote File Access syntax compatible with Shared Resource Management functions. This networking solution will provide a sophisticated, high performance transport function for the new services. The Ethernet connection will be provided first by a Network Interface Unit box connected by HP-IB and later by an HP-IO card using VLSI technology (and compatible with both Ethernet and IEEE 802). Later in 1983, the HP 9000 will have network connections to HP 3000 HP 1000, and HP 9826 family machines as part of an HP Distributed System Network enhancement. Plans for 1984 include a Virtual Terminal capability (remote log on), Remote Data Base Access to HP 9000 Image Data Bases, and Electronic Mail.

Also in 1984, the HP 9000 family will be able to use the new services to other HP 9000 systems or other HP systems over remote data communications links. This includes point to point switched and leased lines using the telephone network or the new Public Data Networks which use the X.25 packet switching protocol. This extends the CAE applications beyond the single facility, local clusters into a worldwide HP compatible network.

Shortly after introduction (1H 1983), the HP 9000 systems, both HP-BASIC and HP-UX, will be connected to the Shared Resource Management System using the 700,000 bit per second Resource Management Link (introduced in July, 1982). This allows HP9816/26/36/35/45 desktop computers and HP 9000 workstations to share discs, printers, and plotters. This connects a wide range of HP-BASIC desktop workstations and provides an upgrade path for existing 9835/45 customers.

NON-HP NETWORKS

In addition to HP networks, HP 9000 systems will provide users access to their existing central mainframes. The first step is the Asynchronous Terminal Emulators and Unix support mentioned earlier. These provide a low cost, flexible connection for timesharing and low volume data and program transfer. For higher performance and data integrity, specific solutions will be provided for IBM and Digital Equipment's VAX mainframes.

In the 3rd quarter of 1983, an IBM 2780/3780 emulator will be available on the HP 9000 with the HP-UX operating systems. This provides batch access to larger IBM mainframes for data and program transfer and job submittal. The features will be the same as the 9845 RJE emulator. For IBM systems requiring interactive access, an IBM 3270 terminal emulator will be developed by the second half of 1983 (Bisync protocol). The 3270 emulator will be similar to HP's Interactive Mainframe Link 3000 product and will be developed using the extensive test facilities at Information Networks Division. IBM System Network Architecture connections will be considered as CAE applications dictate.

To access data and programs on DEC VAX mainframes, often found in large R&D organizations, H-P will develop a File Transfer capability. If necessary HP will implement a small part of DECnet, but prefers that DEC and other vendors move towards industry or international standard protocols. The plan for a File Transfer capability would include support of multiple data communications links: remote, local area networks, and public data networks (X.25).

OEM PRODUCTS

In addition to specific solutions for HP and non-HP systems, the HP 9000 will provide general data communications products. First, in the second half of 1983 a set of tools for developing custom data communications products on the Programmable Serial Interface communications card. The customer will be able to develop protocols on the microprocessor based card and have access to it through standard operating system procedures. Second, after providing HP networking products on the X.25 and 10 mega-bit base band networks, customers will be given access to these link level standards, for developing proprietary networking products or interfacing to non-HP mainframes. These capabilities will give added flexibility for OEMs, 3rd parties, and volume end user customers.

GROWTH PATHS

A very important part of the HP 9000 networking strategy is the growth paths for initial products such as the Shared Resource Manager and the initial Ethernet network capability. The SRM will include increasing functionality such as a gateway function. This will allow a cluster of workstations to access a larger network through the SRM and in the future access additional network functions. The Ethernet network will be upgraded to full IEEE 802 compatibility using an HP-IO interface card (the Network Interface Unit box will also be capable of being upgraded to IEEE 802). The initial user services will be expanded to connect to other HP systems (1000, 3000, 98x6) and to provide increasing performance and functionality. The evolution of the HP 9000 network capabilities is a key part of the Engineering Productivity area of MPN.

NETWORKING STANDARDS

Finally, Hewlett-Packard supports the current efforts to standardize networking links and protocols as a fundamental part of HP Distributed System Network strategy. HP 9000 networking products will be implemented using the International Standards Organization Open System Interconnect (seven layer) Model, which provides future flexibility for adopting available standards (eg. X.25 packet switching). HP has recently been a leader in bringing together the specifications of IEEE 802 and Ethernet, to insure there is only one 10 mega-bit base band network in the future. HP's commitment to intervendor standards is evidenced by the recent agreement of European Computer Manufacturers, including HP, to push for international standards for networking software on Ethernet.



HP 9000/HP-UX

HP-UX OPERATING SYSTEM

FEATURES

BENEFITS

EXTENSIVE I/O REDIRECTION

EASE IN:

TESTING AND VERIFYING
SOFTWARE

POWERFUL COMMAND
INTERPRETER

REUSING SOFTWARE

HIERARCHICAL FILE
SYSTEM

MODIFYING SOFTWARE

MODULAR PROGRAMMING
ENVIRONMENT

OPPORTUNITY FOR EASY GROWTH
OVER A WIDE RANGE OF
COMPUTERS

BASED ON HIGHLY
TRANSPORTABLE UNIX
SOFTWARE

HP-UX OPERATING SYSTEM

A UNIX OS FOR THE ENGINEERING
ENVIRONMENT

HP-UX INCLUDES:

UNIX SYSTEM III with C COMPILER
VIRTUAL MEMORY
HP FORTRAN 77
HP PASCAL
DGL/AGP GRAPHICS
IMAGE DBMS
HP DATA COMMUNICATIONS
HP PERIPHERAL SUPPORT

HP-UX: A New Direction for HP Software

In 1980, the Hewlett-Packard Computer Group announced a new campaign called the Manufacturer's Productivity Network. Although both customers and HP marketing organizations recognize the tremendous value of such a concept, Hewlett-Packard has been slow in developing the tools necessary for the network. The HP-UX Operating System, a derivative of Bell Laboratories' acclaimed System III Unix, is the result of a cooperative effort among divisions of the Computer and Personal Computer Groups to define one software system with an operating system, libraries, and compilers which will be supported by all HP Computer families. The implementation of this product is a significant step toward the successful realization of MPN.

What is Unix ??

Unix is an operating system developed by Bell Laboratories in the late 1960's which supports multiple users, tasking, and languages. Unix evolved from the frustration of Ken Thompson, a Bell computer scientist, with large, centralized computers and their complicated, restrictive operating systems. Thompson desired to write a simple, straight-forward operating system which could offer the functionality of the large mainframe computer, yet execute on an inexpensive hardware set. The resulting system has been widely recognized in both universities and the commercial marketplace as a highly transportable, productive software development tool.

Within the Unix Operating System, there are two components: the kernel and the file system. The kernel is the layer of software which interfaces directly with the computer hardware and consists primarily of system calls (EXEC calls on the HP1000), I/O device drivers, and memory management routines. The Unix file system is a hierarchical or "tree-like" structure which permits directories to contain pointers to other directories. The Unix file system contains a wealth of "C" software designed to perform program development and system maintenance functions. In addition, there are a variety of application programs available which have contributed enormously to Unix's popularity.

These application programs have evolved under the scrutiny of rigid, modular programming practices. The premise of modular programming is to write short blocks of code which address exactly one function and have only one input, one output, and one entry point. Bell Laboratories has used this programming practice with the "C" programming language design application programs which address a wide variety of problems from character recognition to text preparation to electronic mail.

The Unix - "C" Language Relationship

Another reason for Unix's popularity is that it is highly machine-independent and available on a wide variety of hardware products. This phenomenon can be attributed to the fact that Unix was written in a high-level language called "C". To port Unix to a new hardware set, one only needs to develop a native "C" compiler and a host environment to generate code. This ease in transporting Unix software has resulted in Unix emerging as the standard operating system for 16-bit microprocessors and availability on such vendor's mainframes as DEC, IBM, and CDC.

"C" Language, in terms of data structures and control statements, is similar to Pascal. "C" Language, however, does not have the self-documenting syntax of Pascal and does not rigidly enforce strong type checking. The latter point makes "C" an attractive alternative for system programmers who wish to optimize performance.

Unix: The Primary Features

It has often been said, "The more you know about computers, the more you'll like Unix." Although Unix is used as a learning tool for undergraduates at universities, it takes a seasoned programmer to fully utilize Unix's tools and realize significant productivity gains.

Unix is designed around the concept of the "pure program." This concept promotes the decomposition of programs into modules which address exactly one function, with one entry point, one input, and one output. Unix was designed on this principle and offers tremendous rewards to users who design their application software with this concept in mind. The following four features illustrate some of the power and flexibility of Unix.

Uniform treatment of files, directories, and devices- This is an extension to the concept of unified I/O on the 9800 desktop computers. Access to any file, directory, or device is handled with identical syntax using the simple READ/WRITE commands of the particular language being used. The operating system will detect which of the three categories an object belongs to. This feature provides for loose connectivity in structured software design.

Mirrored capability between commands and system intrinsics- This is an essential feature for development of modular programs. This permits one to open, create, and manipulate files, execute and synchronize programs, and signal programs without having to write programs to do so. The power of the command interpreter, called the shell, will be further highlighted with the next two points.

Redirected I/O-- As mentioned, Unix utilizes functional modules with one input and one output. In writing such a functional program, designers are allowed to declare the input and output to be "standard". When a compiler encounters such a declaration, provisions are made to establish the I/O linkage, but the actual linkage is declared and performed at RUN-TIME. For example, say we have a short, already compiled program named "cube" which takes its standard input, cubes it, and outputs the result to its standard output. In addition, suppose we have a file named "datum" with test data for "cube" and wish to use datum as cube's standard input. Let's type:

```
cube<datum
```

This will result in cube being loaded into the machine, datum being opened and linked to cube, and cube executing until data is exhausted. The output will return to the terminal since it is the default standard output. But suppose we want to send the output to a printer. Since Unix treats all files and devices identically, this is simply done by typing:

```
cube<datum>2631_printer
```

Rather than a printer, one could have substituted a name for a file and the output would have been routed there. If the file name did not already exist, the operating system would have created a file with that name automatically before executing the program. This concept is called redirected I/O and is invaluable for testing and reusing modular code.

Piping of data-- Given redirected I/O, wouldn't it be nice if functional modules could be linked together at run-time so the output of one module fed the input of another module. In Unix, this is called "piping". Say that in addition to "cube", we have a program called "sqrt" which takes the square root of its input and outputs the result. Let's type the following:

```
cube<datum | sqrt >2631_printer
```

Since HP-UX is a multi-tasking system, cube and sqrt each will be loaded into a separate partition in memory and executed on different processors, if available. Again, cube will begin taking data from datum and executing. When cube begins outputting data, however, sqrt will begin execution using cube's output as its input. The two programs will continue to execute concurrently with sqrt's output being directed to the printer. Piping is a valuable tool for using simple programs as building blocks to construct more complex programs.

To summarize, the benefits the programmer will realize by using these tools are listed below:

1. Ease in testing and verification of software
2. Ease in reusing software
3. Ease in modifying software

In Appendix A, there is an example which illustrates how these concepts can be applied.

What is HP-UX ??

There are good reasons why Unix has been slow in gaining popularity in the production and engineering world. Some of these are: no vendor support, questionable reliability, no virtual memory, and primitive data base management and graphics. The intent of HP-UX is to remove all of the above objections without sacrificing compatibility with existing Unix products.

Hewlett-Packard has used AT&T's System III Unix as a standard for determining compatibility with the Unix community. Thus, as a basis for compatibility, HP-UX contains all System III intrinsics and all libraries which are not inherently dependent on another vendor's hardware (ie. some graphics and I/O primitives). Hewlett-Packard has then reviewed the "C" application software available with System III and selected all commands relevant to computer-aided engineering and software development.

The operating system interface is just one consideration for software compatibility and transportability. To insure compatibility among HP products, the HP-UX Engineering Task Force has also specified a standard for the languages and engineering libraries which reside on top of HP-UX.

For languages, HP FORTRAN 77, HP Pascal, and "C" Language have been approved by the Task Force. HP FORTRAN 77 is a robust compiler which fully complies with the ANSI standard and includes additional provisions for an acceptable level of backwards compatibility with FORTRAN 66 and FORTRAN 4X. Pascal on HP-UX will track the current Hewlett-Packard Pascal standard. The "C" Language on HP-UX will comply with the description rendered in Kernighan and Ritchie's text "The C Programming Language."

Graphics on HP-UX will be identical to the DGL/AGP Library which the HP1000 people know as HP1000 Graphics II. With the HP 9000 an alias file will be available with more descriptive names since HP-UX is not limited to 5 character descriptors for library names. In the case of DGL, these new names will be compatible with the HP 9826 Pascal Graphics.

The HP-UX Task Force has adopted IMAGE as its data base management library. Representatives of the product line currently supporting IMAGE have conferred and defined one IMAGE implementation which will be supported on all HP-UX implementations.

The mentioned components of HP-UX represent only the beginning of a software standardization effort for the HP Computer Group. Currently negotiations are in progress for developing standards for data communications, real time and I/O, and virtual memory. (Note: the virtual memory implementation on the HP9000 is tran-

sparent to the user. The VM standard refers to the possibility of transferring control to the user in setting memory management parameters.)

A Word on Compatibility (In terms of the HP 9000)

Since the HP 9000 is Hewlett-Packard's first entry in the 32-bit technical computer market, it is expected that many customers will have concerns over software transportability. The most critical issue effecting transportability concerns the source level compatibility of languages and their associated libraries. This issue is discussed in a series of accompanying white papers covering HP 9000 languages. What follows is a discussion of Unix compatibility.

When discussing Unix compatibility, many users will launch into a discussion of which AT&T "C" Language commands are supported and which are not. This consideration is really peripheral to the real issue of the "C" language compiler and its supported libraries and Unix intrinsics. Since "C" Language is the system programming language in which Unix was written, a good, robust "C" compiler can port any Unix software not included with HP-UX.

Discussing "C" Language compatibility is as nebulous as Pascal compatibility. No standards organization exists and many of the Unix libraries have become a defacto part of the language. As mentioned HP-UX uses a text written by two key Bell Labs designers, Brian Kernighan and Dennis Ritchie, as a guideline for HP-UX compiler design. HP-UX's strict adherence to System III Unix intrinsics and libraries will provide customers the best compatibility with AT&T possible with a non-DEC architecture.

Although the issue of supported Unix commands is a peripheral to the compatibility story, it is none the less an issue. It is important to note that not all the HP-UX commands will be available at introduction. As new commands become available, they will be provided to all HP-UX customers who have purchased a support service. Customers not on support services will receive a discount when purchasing the most current software revision. The HP 9000 HP-UX Technical Brochure contains a list of all commands currently available.

Over the last five years, the University of California at Berkeley has emerged as a driving force in the Unix community and distributes their own version of Unix to run on the DEC VAX. Berkeley's contribution to Unix is to increase the performance of Unix by redesigning the buffering scheme of the Unix kernel and introduce virtual memory on Unix. It is possible that some questions may arise concerning HP-UX vx Berkeley compatibility.

The only common piece of code between Berkeley and HP-UX is VI, a

popular Unix full-screen editor. The HP 9000 design team, however, has spent considerable time studying Berkeley Unix and the Berkeley virtual memory implementation has impacted HP-UX. A current proposal for HP-UX virtual memory extensions will give the HP 9000 the same functionality as the Berkeley implementation. In addition, Berkeley and HP-UX have both incorporated Ethernet with the Arpanet transport into their local area network strategy.

To summarize, no conscious decision has been made to make HP-UX syntactically or architecturally compatible with Berkeley. The natural evolution of HP-UX however, has encompassed much of Berkeley's functionality.

Licensing

HP-UX utilizes software licensed by AT&T to implement the Unix commands and libraries. This means that Hewlett-Packard Co. must comply with AT&T's System III Unix source license. Some of the ramifications of this source license are:

- * Hewlett-Packard has the right to sell binary versions of the Unix code
- * Unix source code cannot leave the USA
(HP will not make source code available, anyway)
- * HP customers cannot resell HP-UX (currently)
- * Every HP customer must sign a customer agreement before a computer will ship.

The first two points are straight-forward. Hewlett-Packard has paid AT&T the necessary \$43,000 to obtain rights to the Unix source code. To insure that adequate laws are available to protect their software copyright, that source code cannot leave the USA. ESO, however, has no plans at this time to sell source code to US customers. This is due to the following:

1. The cost and legal liability will be excessive.
(Each customer must pay AT&T \$43,000, plus something to HP to make it viable for us to support the code.)
2. Both Unix and HP-UX are hierarchical software systems. This means that access to most machine functions are available in a high-level language. We assume that most customers will prefer to work in a high-level language to minimize software in-

vestment and retain software transportability.

The point concerning resale of software is more sensitive. Currently, HP customers cannot resell or transfer HP-UX software. This does not necessarily mean we cannot have HP-UX OEMs. The OEM must either get his own AT&T source license (\$43,000) or be content to let HP license his customers directly (ie. the OEM must report his sales to HP). All parties, including AT&T, recognize that this agreement is not satisfactory and a new Unix OEM agreement is expected from AT&T soon. When the new agreement is available, the field will be informed immediately.

The last point is CRUCIAL to ALL HP SALES REPRESENTATIVES !! Every order for HP-UX must be accompanied by a signed customer agreement. The procedure for the SR will be to include a prepared agreement with every quote on a HP-UX system. This agreement should be signed at the same time and by the same person who signs the customer purchase order. Before the order is transmitted, local HP order processing personnel should have the completed agreement. Failure to follow this procedure correctly will result in factory-cancelled orders, unhappy customers, and embarrassed SR's.

Appendix A

A Case Study: Random Radio Corp.

Our story opens with Roger Tenfore, a young support engineer who has just been hired by Random Radio. Roger's first assignment was to assist Random's quality assurance manager, Ben Tested, evaluate the reliability of a new transmitter design.

Ben has just given Roger input impedance measurements from the first 100 production units of the modulator-amplifier assembly. From these measurements, Ben described an algorithm for estimating the power dissipation of the complete transmitter unit. Random's mechanical engineering staff provided Roger with a model which estimated the temperature inside the transmitter housing based on power dissipation and normal operating conditions. Finally, based on this temperature, Ben's staff had determined a probability distribution which described the likelihood of a failure within the first five years of operation. Roger's assignment was to determine the expected number of units in the first production run which would fail over a five year period. With the reckless abandon of a newly-graduated engineer, Roger quickly sketched the structure in figure 1A and immediately began coding.

Sharing the same office space with Roger, is a veteran engineer named Willy Wunderkind. Willy is really a hip dude who studied structured programming years ago and had long since sworn

off GOTO statements. Willy is also one of those people who makes it his business to know everyone else's business. Thus, after several conversations with Roger, Willy came to understand the reliability problem. That afternoon during a coffee break, Willy observed Roger coding away. So Willy, loving the challenge of a new program, grabbed the back of an envelope and sketched his own interpretation of how the problem should be structured (see fig 1B).

The next day Roger appeared in Ben's office with an arm full of listings. Ben looked up from his desk and remarked,

"Got that program running yet?"

Happy as a puppy, Roger nodded his head and replied,

"Yup!!"

"Great! I'm glad to see you turned it around so quickly."

Then a look of judicious concern swept over Ben's face and he remarked, "Oh Roger, I've been meaning to talk to you about your program." Ben went on to explain that measurements for 50 more units had become available and he would like the results of this run compared to the initial 100 units. Also, the boys over in R&D were thinking about changing the ventilation design of the transmitter packaging and had devised a second temperature model. Ben wanted Roger to determine the impact that the new design would have on the reliability of the product. The "last straw" came when Ben remarked that there had been component vendor changes and that there would be minor modifications to the probability distribution as well.

That afternoon in his office, Roger sulked and complained to Willy about how managers never knew what they wanted and how computers create more work than they perform. Roger showed Willy his listing and complained that between the varying sizes of the production runs and the two different temperature models, he would have to generate several different versions of the same program to get the answer. Trying to look on the bright side, Roger conceded that fixing the probability distribution would be simple after he remembered where in the code he declared it. At this point Willy fetched his version of the program structure and remarked,

"You know Roger, maybe we should choose a program structure which is more closely related to the problem."

Willy went on to explain that his structure more closely observed how data actually flowed through the problem. Roger could have made his program independent of the number of data input items by operating on the data one item at a time until an end-of-file was encountered rather than storing the whole data set in an array. Willy also structured the problem around where the data transformations occurred (ie. impedance to power, power to temperature, etc.) That way if the definition of a transformation changed, one small module could be rewritten without affecting the structure of the program. In the case of the two temperature models, if a machine supports separate compilation, two modules can be written and compiled and then the appropriate module can be linked into the program when needed. Finally, Willy's approach will always produce more readable, and thus maintainable, code.

After pointing out the flaws in Roger's program, Willy took Roger down the hall and showed him Random's new HP 9000 computer. Willy explained how the flexibility of the HP9000's HP-UX operating system would be well suited for Roger's program. Willy and Roger worked together to split up Roger's old program into five smaller programs. The programs are described below:

```
watts- converts impedance into power dissipated
temp1- the first model for converting power dissipation into
temperature
temp2- the second model
prob- converts the temperature into the probability of
failure
accum- accumulates the results and formats them for printing
```

Roger then created two data files:

```
Z100- contained the first 100 impedances
Z50-  contained the additional 50 impedances
```

By executing the following four commands, Roger had the data he needed.

```
watts<Z100 | temp1 | prob | accum > 2631_printer
watts<Z50 | temp1 | prob | accum > 2631_printer
watts<Z100 | temp2 | prob | accum > 2631_printer
watts<Z50 | temp2 | prob | accum > 2631_printer
```

Roger was amazed at how quickly they had derived the results and could not resist taking them to Ben immediately. Ben was impressed with Roger's work and the flexibility of the HP9000.

Later that day Ben stopped by Roger's and Willy's office and asked Roger,

"Roger, I was talking to our computer services manager and he told me they have installed some Hewlett-Packard plotting equipment on that HP9000 system. Do you suppose you could change your program to give us a graphical summary?"

Roger thought for a minute and then replied, "Well, you are only asking me to change one function, so it should be easy. All I have to do is insert a graphics formatter in place of my print formatter and redirect the output to a plot routine."

"Great! I'll be expecting it by tomorrow morning." And with that, Ben walked back to his office. Wasting no time, Roger got right to work on the new report. While Willy broke from his work briefly to give Roger the smile of a proud mentor.

The moral of the story: There is a lot more to HP 9000 HP-UX than just Unix. HP-UX also features the extensions engineers need like virtual memory, AGP graphics, IMAGE data base management, data communications, and HP languages and peripherals. In the past, managers have had to choose between technical computers which support their engineers and those that support their computer scientists. The excitement of the HP 9000 software story is that the 9000 family can satisfy all of a manufacturer's computa-

tional needs.

The End



HP 9000 APPLICATION
SOFTWARE

TCG CAE STRATEGY

1. Provide an alternative to 32 bit, shared systems. Leverage from the desktop heritage in providing a complete range of workstations for individual engineers (MARMOT to DAWN) all configurable in a local area network (EPN).
2. Capitalize on HP's experience in engineering and the R&D investment in engineering applications to provide software tools and solutions for CAE.
3. Motivate software suppliers with key application technologies to move their software to HP CAE computers. Use HP and 3rd party software to build turnkey CAE systems.
4. Gain a competitive advantage with the price/performance of EPN combined with integrated engineering applications and links to Factory and Plant Automation.

9000 SERIES SOFTWARE STRATEGY

1. Continued growing emphasis on solutions in order to enhance the position of our general purpose computer products and identify HP as a supplier of CAE solutions.
2. Develop HP Proprietary Applications where contributions can be made in CAE. (EE, ME & Engineering Management)
3. Be very aggressive encouraging software suppliers who compliment our proprietary programs.
4. Encourage software OEMs to provide vertical applications outside the manufacturing CAE market.

APPLICATIONS STRATEGY

ESO's applications strategy is couched in and supports the TCG CAE Strategy. When discussing the ESO application strategy with customers, it is most clearly presented in the context of MPN with a presentation of the TCG CAE Strategy.

TCG CAE STRATEGY

1. Provide an alternative to 32 bit, shared systems. Leverage from the desktop heritage in providing a complete range of workstations for individual engineers (MARMOT to DAWN) all configurable in a local area network (EPN).
2. Capitalize on HP's experience in engineering and the R&D investment in engineering applications to provide software tools and solutions for CAE.
3. Motivate software suppliers with key application technologies to move their software to HP CAE computers. Use HP and 3rd party software to build turnkey CAE systems.
4. Gain a competitive advantage with the price/performance of EPN combined with integrated engineering applications and links to Factory and Plant Automation.

Threaded through our CAE strategy is the importance of applications. Specified are both proprietary and third-party roles. Clearly the emphasis is on solutions. A summary of the applications strategy for the 9000 series computers follows.

9000 SERIES SOFTWARE STRATEGY

1. Continued growing emphasis on solutions in order to enhance the position of our general purpose computer products and identify HP as a supplier of CAE solutions.
2. Develop HP Proprietary Applications where contributions can be made in CAE. (EE, ME & Engineering Management)
3. Be very aggressive encouraging software suppliers who compliment our proprietary programs.
4. Encourage software OEMs to provide vertical applications outside the manufacturing CAE market.

HP PROPRIETARY PROGRAM

EE PROPRIETARY PROGRAM

The EE proprietary program is managed by ESO and CADL located in California. The most notable contribution to EE CAE is the integration of these applications with a common design data system and user interface, as well as connectivity with other MPN solutions.

The HP 9000 will be the first host of the EE design system. In the future, much of this capability will also be available on the 68000 base computers from DCD.

Before additional detail is explained regarding the EE strategy, it is valuable to list the current buzz names for EE.

HPSPICE: Circuit Simulator to be introduced on the 9000 in November; also available on the HP 1000 today.

DRAW: Lab code name for the schematic capture module of the Logical Design System.

PDS: Physical Design System intro Q3'83.

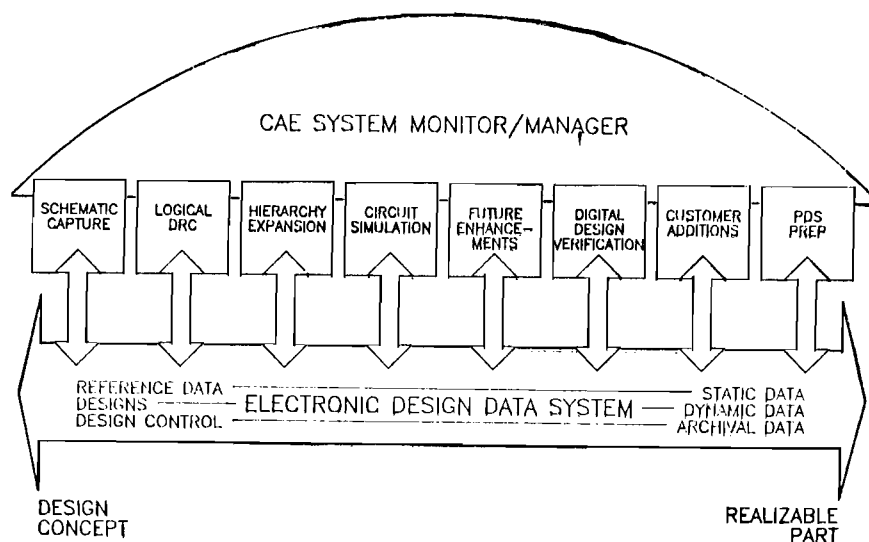
LDS: Logical Design System intro Q3'83.

ESCHER: Lab code name for PDS

NEMBUS: Lab code name for LDS

The EE design system addresses two aspects of electronic design. The logical design system covers conceptual design (through schematics) simulation, analysis, and design output. The physical design system covers the physical implementation of the design, initially in the form of a printed circuit board. The CAE SYSTEMS MONITOR/MANAGER and the ELECTRONIC DESIGN DATA SYSTEM shown in the following models are common between the two systems.

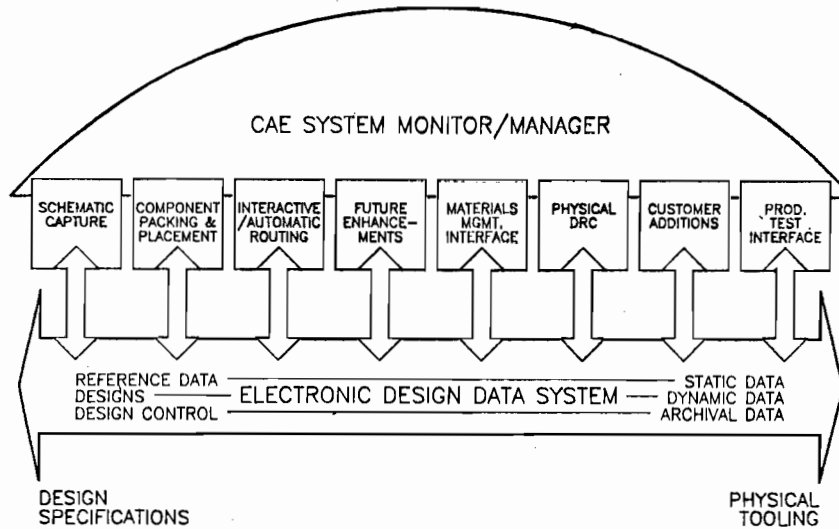
LOGICAL DESIGN SYSTEM



LOGICAL DESIGN SYSTEM

This package of programs aid the engineer who is designing and analyzing an electronic circuit. The schematic capture software is based on DRAW. Design rule checking for digital circuits is available such as input/output consistency, output loading, etc. Circuit simulation is done using HSPICE code. Analysis and verification for digital design will be done with a set of tools including a timing analyzer. Reports such as component lists and net lists will also be available to the engineer and will provide data to the physical design system.

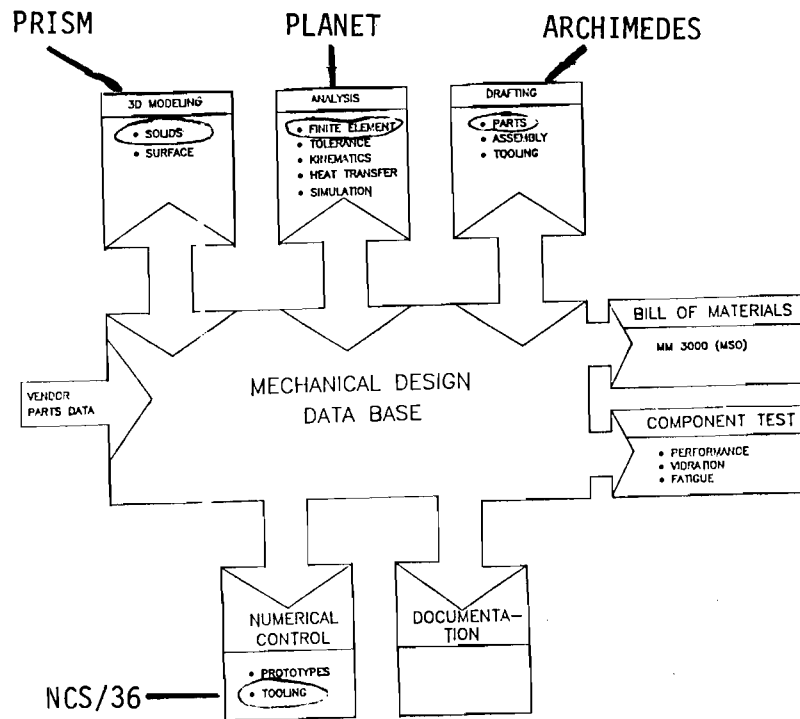
The LDS is scheduled for Q3 of 1983. Future enhancements include a logic simulator as well as utilities and documentation to allow customers to add their own modules to the system.

PHYSICAL DESIGN SYSTEM**PHYSICAL DESIGN SYSTEM FOR PCBs**

The Physical Design System can take circuits designed with the Logical Design System and implement them in the form of PC Board Design. Layout tools include an interactive Editor for manual positioning of gates, components, and traces. Automatic placement of components and automatic multi-layer routing of traces will provide quick turn-around time. On-line design rule checking (DRC) and consistency checking will assure that the PC board meets manufacturing functional requirements. Tooling functions are included such as NC, drill, stitch wire, auto insertion and photo plots. Links are also provided to the Touch-Tone Board Tester (LID) and the MM/3000 Bill of Materials program. Introduction is scheduled for Q3 of 1983. Future enhancements include automatic packing or gates into IC's, as well as utilities and documentation to allow customers to add their own modules to the system.

ME PROPRIETARY PROGRAM

The ME Proprietary Program is managed at BDD located in Germany. Below is a model of the programs required by a mechanical engineer to form a complete CAE solution. The Code names are indicative of the current staffed projects. At this time, there is no common control or shared data base between ME programs, as there is for the EE systems.



Future plans at BDD call for a common data base, and the upgrade translation of the NCS/36 numerical control program for the 9000 products.

In summary, the ME strategic goal is an integrated ME CAE system supporting 3D-Modeling, Finite Element Analysis, Detailed Drafting and 3D Numerical Control.

SWE PROPRIETARY PROGRAM

Two major software activities in the 9000 program are designed to make the 9000 computer family very attractive to customers who are writing their own software.

The first major SWE activity is the implementation of HP-UX. This project is managed out of the ESO lab located in Fort Collins. At introduction over 65 shell commands will be available.

The second major SWE activity is the Graphics 9000 Products being designed by CADL in California. These products can be purchased two ways:

DGL = Device Independent Graphics Language

AGP = Advanced Graphics Package

DGL can be purchased separately and provides low level graphic commands with the advantage of HP supported drivers for future products which require no re-coding of the customer program. The whole package GRAPHICS 9000 consists of DGL and AGP. GRAPHICS 9000 is a superset of the GRAPHICS 1000-II currently being sold on the HP 1000 products. At HP 9000 introduction GRAPHICS 9000 will be available and GRAPHICS 1000 will be enhanced to have comparable capability.

There is general compatability between GRAPHICS 9000 and GRAPHICS 1000, however, 17 commands depend upon differences in the RTE and HP-UX operating systems.

HP PROPRIETARY SOFTWARE

	1982				1983				1984			
	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4		
EE		HP SPICE			PHYSICAL DESIGN LOGICAL DESIGN			IC DESIGN				
ME		ARCHIMEDES PLANET				PRISM						
SWE UTILITIES		UNIX UTILITIES BASIC UTILITIES 9845 TRANSLATOR TERMINAL EMULATOR	AGP/DGL		SCCS* RJE/BISYNC EMULATOR	3270 EMULATOR						
MGT			VISICALC* CRAYONS*			PROJ. MGT* MATH* STAT* PIGPEN						

*Not Staffed

THIRD-PARTY

Major Third-Party software suppliers, in the CAE area, will complement HP's proprietary software. Large existing customer bases and well-known software products make these software houses important in HP's overall solution strategy on the HP 9000.

Additionally, the choice of the HP 9000, as a host for major software products, attests to the power, performance, and system features of the HP 9000 family of computers. The 32-bit customer base is currently using these major products on competitor's machines. The greater offering of Third-Party products will decrease customer conversion time and increase acceptance of the new HP 9000 system for non-HP customers.

Therefore, an aggressive Third-Party program, using Engineering System Division resources and the HP Field, is in place for the HP 9000 family of computers. A North American Direct Mail campaign, timed with the HP 9000 introduction, is being sent to major software suppliers in the EE, ME, and SWE vertical markets. Also, a selected list of software suppliers have been contacted by ESD, so that the software will be converted and available as soon after introduction as possible.

The HP+ Program for Third-Party software is in place for ESD. HP+ program guidelines are being evaluated and updated for all of the Computer Group to make it even more attractive for a third-party to join HP+. Areas under discussion include the development system discounts, incentives for merchandising, and support considerations.

HP has the program in place to attract the Third-Party suppliers, cut their conversion time/costs, and the most attractive price/performance 32-bit engine to drive their software. Software suppliers are of major importance to the HP 9000 CAE solutions offering and the Technical Computer Group overall strategy.

SOFTWARE	DESCRIPTION	MANUF. RELEASE	INTROD.	LANGUAGE
<u>ELECTRICAL ENGINEERING</u>				
HPSPICE	General purpose circuit simulation program for the design & test of analog circuits or analog or digital IC's.	3Q82	1Q83	FORTTRAN & PASCAL
DRAW	Interactive schematic capture program. Supports a hierarchical input structure for both simulation & PCB layout programs.	3Q83	3Q83	PASCAL
LOGICAL DESIGN SYSTEM	Integrated software system for the logical design of electronic systems at the board, box & system levels. Includes both HPSPICE & DRAW with an integrated data base.	3Q83	3Q83	PASCAL
PHYSICAL DESIGN SYSTEM	Integrated software system for the physical layout of printed circuit boards. Supports manual gate packing, manual & automatic routing and placement. Generates photo plots NC tooling, and auto-insertion data.	3Q83	3Q83	PASCAL
IC DESIGN	Graphics based software system for the design & analysis of gate array, standard cell and custom integrated circuits.	2Q84	2Q84	PASCAL
<u>MECHANICAL ENGINEERING</u>				
ARCHIMEDES	A 2-D design & drafting system for the design of mechanical parts and assemblies. Supports variation design applications and personal drawing storage.	4Q82	1Q83	BASIC
PLANET	General purpose finite element analysis program for linear elastic static analysis, linear dynamic model analysis & linear heat conduction.	4Q82	1Q83	BASIC
PRISM	A three-dimensional solids geometric modeling program for mechanical design. Supports line-vertex, swept volume & solids primitives. Also provides detailed drafting support.	4Q83	4Q83	BASIC

SOFTWARE	DESCRIPTION	MANUF. RELEASE	INTROD.	LANGUAGE
<u>SWE/UTILITIES</u>				
UNIX UTILITIES	A set of programming utilities for the HP-UX operating system covering system access, backup, compilers, file system control, document & text preparation, file access, system maintenance & status, system intrinsics.	1Q83	1Q83	C
BASIC UTILITIES	A set of programming utilities for the SUN-BASIC operating system covering file backup, language incompatibilities, file comparisons, profiler, & cross-reference.	4Q82	1Q83	BASIC
AGP-3/DGL	DGL is a set of device-independent subroutines for 2-D graphics. AGP-3 uses DGL as a basis & provides subroutines for 3-D extensions & graphics input device support.	3Q83	1Q83	FORTRAN
9845 TRANSLATOR	Translates basic statements from 9845 programs into Dawn Basic & flags those with no equivalent function.	4Q82	1Q83	BASIC
TERMINAL EMULATOR	Allows the 9000 to operate as a remote virtual terminal. Supports 50 to 9600 baud data rates full or half duplex, EBCDIC & local language character sets.	4Q82	1Q83	UNIX
RJE/BISYNC EMULATOR	Allows the 9000 to communicate with a remote computer & other devices that support IBM 2780/3780 Remote-Job-Entry data transmission terminals.	2Q83	3Q83	UNIX
SCCS	Set of commands to control changes to modules of text-files of source code, documentation, data etc.	3Q83	3Q83	UNIX
3270 EMULATOR	Allows the 9000 to communicate with computers & devices supporting IBM 3270 data transmission terminals.	Not Staffed	*84	UNIX

SOFTWARE	DESCRIPTION	MANUF. RELEASE	INTROD.	LANGUAGE
<u>MANAGEMENT</u>				
VISICALC	Electronic worksheet which allows the users to create tables which can be modified in real-time. Entries can be defined as constants or functions of other variables.	NOT STAFFED		PASCAL
CRAYONS	Used for the generation of graphic presentations for overhead transparencies & management reports. Supports line chart, bar chart & pie chart output.	NOT STAFFED		BASIC
PROJECT MANAGEMENT	Provides critical path analysis for projects with up to 1800 activities for CPM & PERT techniques & 1400 for MPM. Supports GANTT chart output.	NOT STAFFED		BASIC
MATH/STAT	Conversion of the math & stat libraries now available on the 9845. Includes regression analysis, stat graphics, general stat, non-linear regression, monte carlo simulation, AOV, factor analysis & numerical analysis routines.	NOT STAFFED		BASIC

HP 9000 GRAPHICS

HP 9000 Graphics Libraries

BASIC Language System

3D Graphics: BASIC statements
 2D and 3D capability
 device independent I/O
 similar functionality to Siggraph Core

HP-UX System

Graphics/9000 : Translation of Graphics 1000-II
 for HP 9000 hardware
 Descriptive subroutine synonyms

DGL : FORTRAN subroutines
 Device independent 2-D primitives
 Building block

AGP : FORTRAN subroutines
 3-D capability
 Implemented via DGL
 Similar Functionality to Siggraph
 Core

HP 9000

Display Hardware Comparison

DISPLAY	GRAPHICS	ALPHA	COLORS	SPEED	LIGHT PEN	SOFT KEYS
High Perf. Mono-chrome top (9020b)	12" 560x455 dots (.34mm dot size)	184x149mm; 26 lines x 80 char.	17 grey shades	3000 vect/ sec.	YES	YES
High Perf. volot yop (9020c)	13" 560x455 dots (.34mm dot size)	192x156mm; 26 lines x 80 char.	8 true; 4913 dithered	3000 vect/ ec.	YES	YES
Color Video Inter-face (FY83)	19" 512x512 dots (1:1 aspect ratio) 576x455 dots (3:4 aspect ratio)		8 std; expand. to 16 of 4096	1000 vect/ sec.	N/A	N/A
Graphics Terminals	7" - 14" Up to 512x390 dots	48 lines x 80 char.	some Black/ white; some 8;	Up to 300	NO	Some
Vector Display (FY83)	14" - 21" 1K x 1K resolution		Mono chrome	6000	NO	NO

GRAPHICS ON THE HP9000 - SERIES 500

The following paper was developed to provide you with additional information about the HP9000 Computer Family graphics capabilities. This paper discusses the following:

THE HARDWARE

- Graphics Displays for the HP9000 Series 500
- HP9000 Series 500 Display Hardware Comparison

THE SOFTWARE

- BASIC vs. AGP/DGL Graphics

THE HARDWARE

GRAPHICS DISPLAYS FOR THE HP9000 SERIES 500 COMPUTER FAMILY

The HP9000 Series 500 offers a range of graphics display devices. These devices consist of graphic tops (for the integrated models), the Color Video Interface with a large screen monitor, the stand-alone high resolution color display, graphics terminals and vector displays. When choosing a graphics display device, the user must analyze the requirements of his particular application, select the mode of communication from the main-frame to the display, decide on a price range and then select the display which best matches his needs.

-- Graphics Tops --

Low Cost Color Top

The Low Cost Color Top for the integrated versions of the Series 500 computers is designed for the price sensitive customer. These users need color, but don't require the enhanced performance color top. The low cost color top is designed for data display and CAE applications.

High Performance Color Top

The High Performance Color Top for the Model 20 offers very good quality color images. The enhanced speed and increased resolution of this display allows the Model 20/High Performance Color combination to address some design graphic application areas where color is required. EE and ME design activities such as some Schematic Entry, smaller PCB Design, IC Design, 2D Drafting, and 3D Geometric Modeling where a large screen is not essential, are target application areas. The light pen and softkeys featured on this display provide enhanced interactivity and human interface for user ease-of-use applications.

High Performance Monochrome Top

The High Performance Monochrome Top for the Model 20 provides users not requiring color, with enhanced speed and resolution. In place of the color capabilities, the monochrome top offers 17 shades of gray along with the light pen, softkeys and performance available on the high performance color top. The price is comparable to that of the low cost Color Top. Advanced simulation, test, computational engineering tasks and monitoring applications requiring high performance, but not color graphics, are the primary market areas. This display will also handle some design graphics tasks where color or a large screen are not required.

-- Color Video Interface and Monitor --

The Color Video Interface plugs directly into an I/O slot of any computer. Standard RS-343 Red, Green Blue and sync signals are output to communicate with one of several video monitors available from outside vendors.

By providing an inexpensive means of connecting to a large screen monitor, the Color Video interface allows the Series 500 computers to be used in a wide range of applications where a color large screen for output is a must. Many EE and ME design tasks fall into this category as do training activities that require a standard large screen for viewing by a larger audience.

The 16 colors are user selectable in order to provide the programmer with the flexibility required in more sophisticated graphics applications. By supporting a number of vendor monitors, the user can purchase a display with the features he needs and in the price range he identifies to satisfy his own application requirements. For color hardcopy, the Color Video Interface provides a means for interfacing to some of the various color cameras and printers currently on the market.

-- High Resolution Stand-alone Color Display --

Many design - intensive applications require color, high resolution, a larger screen and substantial graphics performance. Our stand-alone color display addresses these applications which cannot be handled by any of the other graphic display alternatives. With performance comparable to that of the high performance color top, this display offers higher resolution, a larger screen, a color map of 256 displayable out of 16 million possible colors and the same friendly user interface via the graphics statements of the BASIC operating system. These features allow it to be used in complex data display, CAE and most CAD applications.

-- Graphics Terminal --

Another type of graphics display device which can be used with the Series 500 Computer Family are Graphics Terminals. Most of HP's graphics terminals are supported now or will soon be on the Series 500 Computers.

Graphics terminals are usually slower than the graphics tops of the Model 20 or the Color Video Interface/Monitor combination, but they provide the traditional mainframe/terminal environment in which many users are accustomed. In the Unix environment, the AGP/DGL packages are available to support the graphics terminals in the popular programming style and device independence previously available only on the HP1000 family.

Graphics terminals usually are involved in application tasks mainly concerned with displaying data. Test, simulation, analysis and program development results are all popular display graphic tasks for graphic terminals.

-- Vector Displays --

Vector Displays produced at Hewlett-Packard's Colorado Springs Division round out the display offering for the HP9000 Computer Family. Targeted for applications not requiring color, but rather high speed and high resolution graphics, these vector displays allow our computers to handle application tasks not addressable by any other display device. Some of these application areas include detailed printed circuit design, ME tolerance drawings, ME part interaction simulation and some animation.

Vector displays produce crisp, clear lines, and because of the technology involved, offer speed not possible by raster devices. Some users will choose to give up the color offered by raster displays in order to take advantage of the increased speed and resolution offered by the vector technology. The electromagnetic technology of the Colorado Springs vector displays also eliminates the previous large price differential between vector and raster, thus making the vector display a much more viable alternative.



HP9000 SERIES 500 DISPLAY HARDWARE SPECIFICATIONS***

DISPLAY	GRAPHICS SCREEN	ALPHA SCREEN	COLORS	SPEED	LIGHT PEN	SOFT KEYS	AVAILABLE	PRICE*	BASIC SUPPORT	UNIX - AGP SUPPORT
Low Cost Color Top	12" 512x390 dots 60 Hz. non-interlaced (.42mm dot size)	217x165 mm 26 lines x 86 char.	Color Map 16 true from 4096	1000 vect./ second	No	No	March '83	\$6K	Yes	Yes
High Performance Color Top	13" 560x455 dots (.34mm) 60 Hz. non-interlaced	192x156 mm 26 lines x 80 char.	Dithered** 4913; 8 true	3K-5K vect./ second	Yes	Yes	Nov. '82	\$20K	Yes	Yes
High Performance Monochrome Top	12" 560x455 dots (.34mm) 60 Hz. non-interlaced	184x149 mm 26 lines x 80 char.	Dithered** 17 shades of gray	3K-5K vect./ second	Yes	Yes	Nov. '82	\$7K	Yes	Yes
Color Video Interface with Monitor	19" 512x512 dots (1:1 aspect ratio); 576x455 dots (4:3 aspect ratio) 60 Hz. non-interlaced	None "Soft Plotter" only	Color Map 16 true from 4096	1000 vect./ second	No	No	Jan. '83	\$10K	Yes	?
High Resolution Stand-Alone Color Display	19" 1024x768 dots 60 Hz. non-interlaced; 736x552 dots 30 Hz. non-interlaced	-	Color Map; 256 from 16 million	3K-5K vect./ second	No	No	Mar. '83	\$25K- \$30K	Yes	May '83
Graphics Terminals	9" - 14" approx. 512 x 390 average	Terminal Dependent	Some B/W and some color	Terminal and interface dependent	No	No	Nov. '82	\$1K- \$20K	Some	Some
Vector Display	14"- 25" 1Kx1K	Vector drawn	Monochrome	6K-8K	No	No	Nov. '82	\$8K- \$20K	?	Yes Planned

*Estimated Price without Mainframe

**Refers to colors created by dot patterns rather than color intensity electronics.

***All specs approximate and plans projected at this time (9/82)

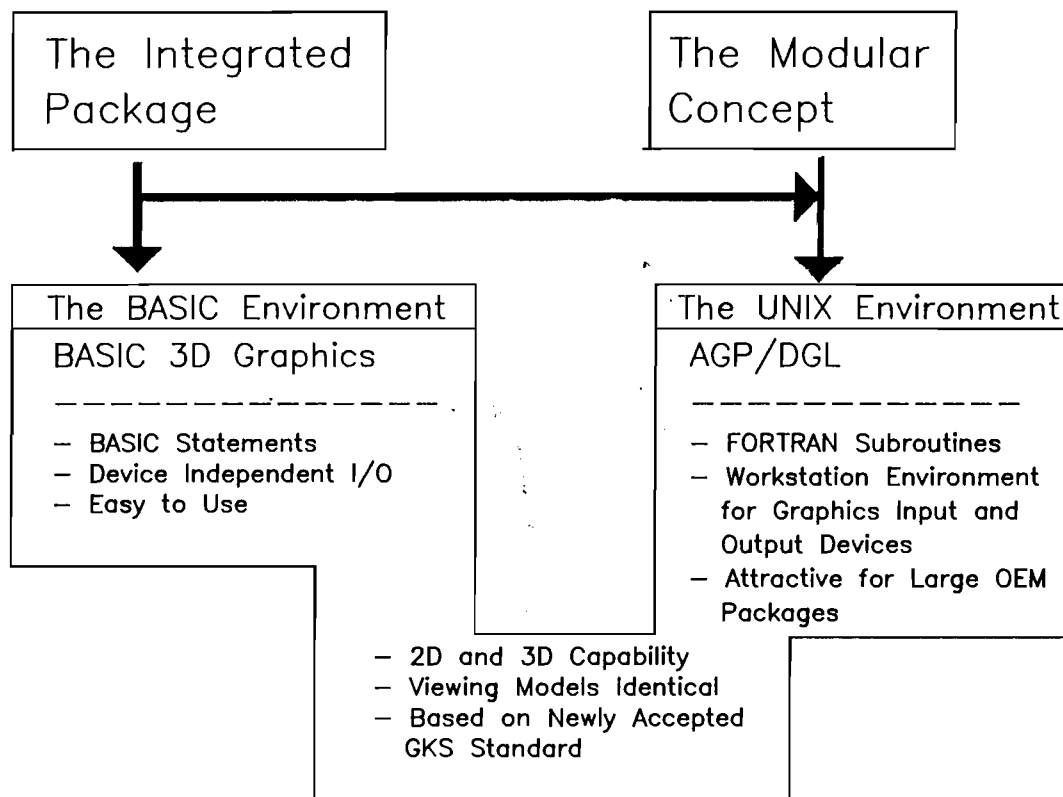
THE SOFTWARE

BASIC vs. AGP/DGL GRAPHICS

Because of the two environments available on the HP9000 computer family, it is necessary to understand the graphics capabilities of each.

The illustration below graphically shows the packaging schemes, the environments and then the graphics systems with their different and common features.

GRAPHICS SOFTWARE SYSTEMS



In summary, the BASIC system and AGP/DGL systems differ very little. Rather, the biggest difference is in the IMPLEMENTATION of the systems.

The viewing models are identical. Of course, since the Basic graphics System is really part of the Basic Operating System (providing actual graphics statements) and AGP/DGL is a separate package of callable subroutines on top of the operating system, the methods for invoking the graphics capabilities of the two systems differ.

The Basic system promotes the same friendliness and ease-of-use present in the 9800 Desktop Computer family. Designed with the end user in mind, the Basic Graphics System is integrated right into the operating system. The user does not have to do anything special in order to have the graphics statements available to him. This integration allows the system to perform much of the environmental set-up tasks for the user. Default values are automatically provided.

AGP is a graphics package written in Fortran with graphics routines callable from Fortran, Pascal, C, or any host language that can call Fortran subroutines. AGP default values set the user up in a 2D environment. It's subroutine library implementation appeals to large OEM's who can incorporate the routines into their own systems when developing a large software package.

In the past, graphics systems were evaluated against the CORE standard for lack of any other comparison norm. The problem with CORE is that its specification was limited in some areas. When AGP was first developed on HP's 1000 family of computers, it deviated from the CORE standard somewhat in order to incorporate features and flexibility not outlined in the standard. During the summer of 1982, the ANSI committee voted to accept GKS (Graphics Kernal System) a European standard developed in Germany, because of its extended capabilities. We were instrumental in this decision; that's because we had already designed AGP/DGL to match GKS very closely. Our foresight paid off - we already have a product that adheres to the specs of the new standard.

Then, when our designers developed the BASIC graphics for the Series 500, they patterned it directly after the existing AGP system. It's flavor was changed to accomodate the desktop computer oriented user, but it's functionality matches that of AGP/DGL including the subsequent enhancements added since the initial GRAPH 1000 package.



HP 9000 I/O

1000/9826/9000 I/O COMPARISON

1000 9826 9000

BANDWIDTH	MEDIUM TO HIGH	MEDIUM	HIGH
TRANSFER RATE SHORT BLOCK LONG BLOCK	MEDIUM	HIGH	MEDIUM
	MEDIUM	MEDIUM	HIGH
INTERRUPT RESPONSE TIME	GOOD (best with privileged I/O)	BETTER	GOOD

HP 9000 I/O SUMMARY

SIMILAR IN CONCEPT TO 9826/36,
BUT ENHANCED BY MULTIPROGRAMMING.

VERY FAST LONG DATA TRANSFERS

MULTIPLE SIMULTANEOUS DMA

MODEST PERFORMANCE FOR SHORT
TRANSFERS.

IOP'S OFFLOAD CPU'S FOR
COMPUTATION.

HP 9000 I/O WHITE PAPER

The intent of this paper is to give the reader an understanding of the HP 9000 I/O features and performance to the level a customer would require to make a purchase decision.

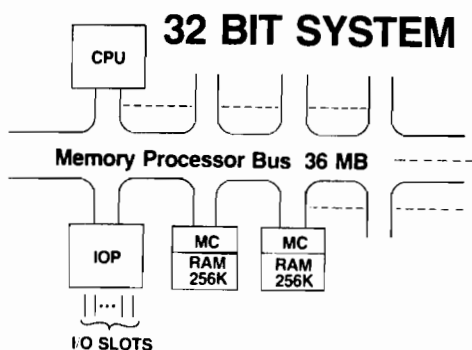
I/O System Overview

I think you would agree that the I/O system of an HP 1000 need be different from that of an IBM mainframe, each designed to best meet the needs of its target market. The HP 9000 design center is CAE and scientific computation. A secondary market is CAT. The architecture of the HP 9000 is designed to offload I/O operations from the CPU, allowing the CPU to concentrate on computation. Additionally, the architecture supports many high speed DMA channels operating concurrently at high data rates, providing excellent system throughput.

Both operating systems of the HP 9000, HP-UX and the BASIC Language System, are designed with the target market in mind. The BASIC Language System has a powerful set of I/O statements and features for file I/O and event-driven interrupt handling. HP-UX currently is a file I/O system only. Capabilities planned for later release include real-time intrinsics and low level access to I/O cards via C language.

I/O Hardware Description

The HP 9000 architecture consists of the CPU, IOP (I/O Processor), and Memory, tightly coupled on the Memory Processor Bus, or MPB.



The IOP is nearly the complexity of the CPU chip itself, and provides the facilities for offloading the CPU from I/O operations. The IOP has 8 high-speed DMA channels. A second and third IOP can be added onto the MPB, effectively tripling the number of DMA channels and I/O bandwidth. The benefit of this architecture is high system throughput in computational systems having several discs and other high load I/O devices. The price paid in offloading the CPU is the time necessary to set up the IOP for each transaction, this time being a factor only for transfers of a few words.

The device adaptor (i.e. I/O card) offering is as follows:

GP-IO

- * 16 bit data paths plus status/control
- * All functions programmable
- * ~ 1 Mb/sec data rate

HP-IB

- * Dual speed implementation, high speed subset protocol for CS-80 discs, and full protocol lower speed for other peripherals and instrumentation.
- * 1 Mb/sec high speed
- * 700 kb/sec in normal mode

ASI (Asynchronous Serial Interface)

- * RS-232 full or half duplex
- * Full modem controls
- * BAUD rates up to 19.2k

MUX

- * 8 channel terminal multiplexer

PSI (Programmable Serial Interface)

- * Supports several protocols for advanced networking
- * Will be user programmable
- * 800 kbits/sec.

I/O System Software Description

HP-UX, as stated earlier, is currently a file oriented system supporting a range of discs, printers, plotters, terminals, displays, and data tablet. Planned enhancements include real-time intrinsics and low level access to I/O cards via C language.

The BASIC Language System is a proper superset of the 9826/36. 9826/36 BASIC programs run with minimal modification. No translator is needed. Several features exist in HP 9000 BASIC not found on the 9826/36, most to take advantage of the high bandwidth available. Event-driven interrupt programming, true overlapped I/O, interprocess communication, and circular buffers are a few of the features offered. These features, coupled with multi-programming and CRT "screening", allow very complex systems to be developed consisting of multiple intercommunicating programs monitored in separate user-defined "screens" on the CRT.

Compatibility with the 9845 is not as straightforward. Our experiences with porting show that it is easy if the code was designed with porting in mind and if mostly file I/O. It is moderately difficult with complex I/O, and is hard if done in assembly language. More information can be found in the I/O compatibility white paper.

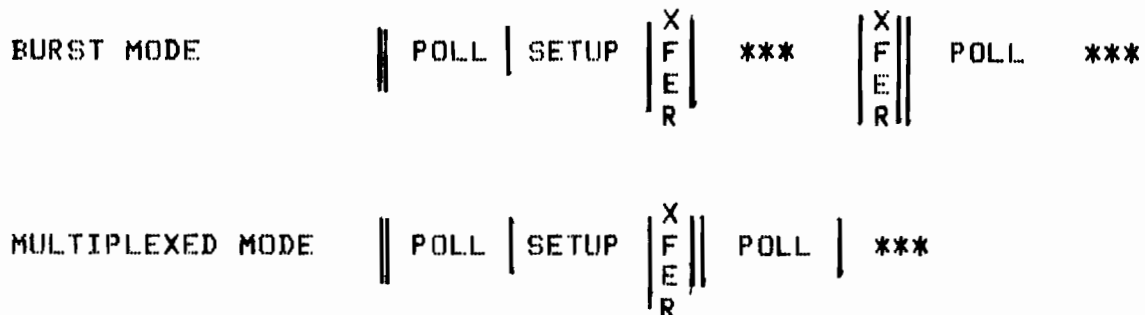
Characterization of the HP 9000 I/O Performance

At the time of this writing work was continuing on the IOP, I/O cards, and tuning of the BASIC and HP-UX systems. Therefore, characterization of the I/O system was incomplete. The following discussion will provide a framework for later characterization.

Some understanding of the IOP and I/O card operation is required. The IOP has a burst mode and a MPX(multiplexed) mode. I/O cards are designed to work in one mode or the other. The mode used by the IOP is then determined by the implementation of the I/O card.

A data transfer in the MPX mode begins when a request from a MPX mode card is recognized by the IOP during a POLL cycle. Some setup is then required, followed by the transfer of one word(two bytes). To transfer a string of information this same POLL, SETUP, XFER, sequence must be repeated many times. The transfer rate is then determined by the length of time required to perform this sequence. The maximum transfer rate in the MPX mode is 485 k transfers/second, or 970 kbytes/sec. This is independent of whether all transfers are on one DMA channel, or multiplexed across all 8 channels. For example, 2 channels could each operate concurrently at over 400 kbytes/sec. The GP-IO card is a MPX mode card, and has a maximum data rate of 970 kbytes/sec, as expected.

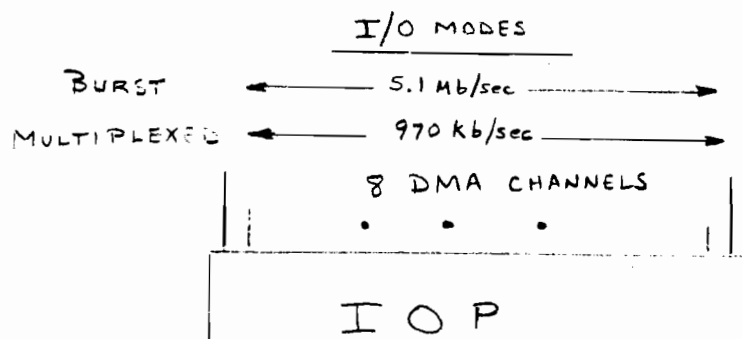
I/O MODES



A burst mode I/O card has the advantage of being able to have multiple transfers for each POLL, SETUP sequence, and furthermore, has a shorter XFER cycle. Up to 32 XFER cycles can be requested per POLL, SETUP sequence resulting in a maximum transfer rate of over 5 Mbytes/sec. As with the multiplexed mode, this is independent of whether a series of burst transfers are on one DMA channel, or multiplexed across several channels at each IOP POLL.

The HP-IB card is a 6 XFER burst card. The maximum rate for a 6 XFER card is 1.6 M transfers/sec. The HP-IB card being a byte transfer card, it thus has a data rate of 1.6 Mbytes/sec. In actuality this card averages 1 Mbyte/sec, doing 1.6 Mbyte/sec bursts with dead time in between when other operations are carried out by the card. Currently no cards are planned implementing a burst length longer than six.

As you can see, the IOP can be consumed by a single MPX mode card running continuously at 970 kbytes/sec, or a burst mode card at 5.1 Mb/sec. This points out that it is the percentage of IOP bandwidth consumed per I/O card rather than the kbytes or Mbytes per second. For example, a GP-IO card at 200 kb/sec would be consuming ~20% of available bandwidth, whereas an HP-IB card at the same 200 kbytes/sec only about 12%.



Now the basis is established for determining the HP 9000 I/O performance in a given application. Left to consideration are the duty cycles of each I/O device in the system. Rarely are devices going continuously, i.e. 100% duty cycle. Also, up to now we have discussed the performance of a single IOP. A second, and third IOP can be added, multiplying the available bandwidth. Two heavy demand I/O devices requiring fast servicing could, for example, be on separate IOPs rather than competing for the resources of a single IOP.

Benchmark Comparisons

The following tables contain performance comparisons of the 9825, 9826, 9845, and 1000 computers in simple I/O benchmarks for the GP-IO and HP-IB cards. These benchmarks were done on the Model 20 (DAWN) with the BASIC Language System, and were done before final tuning of the IOP, I/O cards, and software system. The table lists the ratio of the HP 9000 performance to the other systems, where the 9000 is faster where the ratio is greater than 1.

Performance of device I/O on Mikkelson
Benchmarks to REAL Source/Sink Devices
(Not to test box or self test jumpers.)

TABLE

Ratio of DAWN performance to other machines

GPIO (Bigger $\implies$ DAWN Faster if >1)

Machine	OUTPUT REAL	ENTER 1 REAL	ENTER K STRING	OUTPUT 1K STRING	TRANSFER 17 BYTES	SIMPLE AVERAGE
9825	1.37	1.74	7.78	4.86	1.07	3.32X
9826 Basic	.98	.93	8.14	3.05	----	3.28X
9826 HPL	1.21	1.11	1.23	1.84	.97	1.23X
9845B	8.04	6.88	7.95	7.23	1.28	6.24X
9845/200	7.70	5.47	8.44	7.87	1.32	6.12X
1000F FORTRAN	2.58	4.87	3.55	1.92	----	3.18X
DAWN Time (millisec)	2.33	2.25	4.88	5.37	4.88	----

HPIB - HPIO-HPIB

	OUTPUT REAL	ENTER 1 REAL	ENTER K STRING	OUTPUT 1K STRING	TRANSFER 17 BYTES	SIMPLE AVERAGE
9825	.90	.80	.15/5.48	3.45	1.08	1.27/2.34
9826 BASIC	.83	.43	.15/5.72	2.28	----	.88/2.28
9826 HPL	.79	.51	.02/.88	1.38	.97	.73/.90
9845B	5.24	3.08	.15/5.6	6.53	1.27	3.25/4.34
9845/200	5.01	2.51	.16/5.9	6.15	1.32	3.03/4.18
1000F FORTRAN	1.88	2.14	.067/2.40	1.42	----	1.32/1.93
DAWN Time (millisec)	3.57	4.90	259/894	7.25	4.72	-----

Note that for longer data transfers, the overhead associated with setting up the IOP becomes insignificant, and the system performs very well. For short transfers and for ENTER statements requiring a terminate on character match, performance is less than the 9826/36.

A data logging benchmark was done with the BASIC Language System utilizing a circular buffer. The circular buffer is simply comprised of two TRANSFER statements, one into, and one out of, the same buffer. Fill and Empty pointers are automatically updated. A GP-IO card was used for data input, and the data logged onto a 7912 disc. The continuous data rate was over 400 kbytes/sec, and was done before final tuning of the I/O system.

Comparisons with the competition, such as the VAX family have not formally begun. The published data for the VAX 730 lists a maximum DMA bandwidth of only 1.5 Mb/sec, and for the VAX 780, 13.1 Mb/sec. It is expected that the HP 9000 I/O with a single IOP will outperform the 730. Characterization of the second and third IOP is not complete, and until benchmarks have been done, a comparison will not be made to the 750 or 780. It is expected that the comparison will be favorable.

Summary

The I/O hardware and software systems of the HP 9000 are well tuned to the CAE and scientific markets, and will be enhanced further. The software systems sit on top of a high bandwidth, expandable architecture. HP-UX systems can thus handle a large number of discs and other I/O devices with good system throughput. The BASIC Language System I/O statements coupled with multiprogramming can tackle complex I/O applications, positioning the Model 20 as the high end I/O controller.





HP 9000 AS A GENERAL
PURPOSE MINI

THE HP 9000 AS A GENERAL PURPOSE MINI

RELIABILITY:	EXCELLENT EXTENSIVE SELF-TEST
REDUNDANCY:	NO. MULTI CPU SYSTEM IS NOT REDUNDANT HARDWARE
ASSEMBLY LANGUAGE:	NO PLANS. C LANGUAGE A SUBSTITUTE IN SOME CASES
BACKPLANE DOCUMENTATION:	NO PLANS. PSI WILL BE OFFERED. BREADBOARD CARDS POSSIBLE FUTURE PRODUCT.
POWER FAIL PROTECT:	NOT OFFERED. POSSIBLE FUTURE ENHANCEMENT
MICROPROGRAMMING:	NO PLANS.

ASSEMBLY LANGUAGE

AVAILABILITY

The Focus Assembly Language will not be supported through either the BASIC or the HP-UX operating systems.

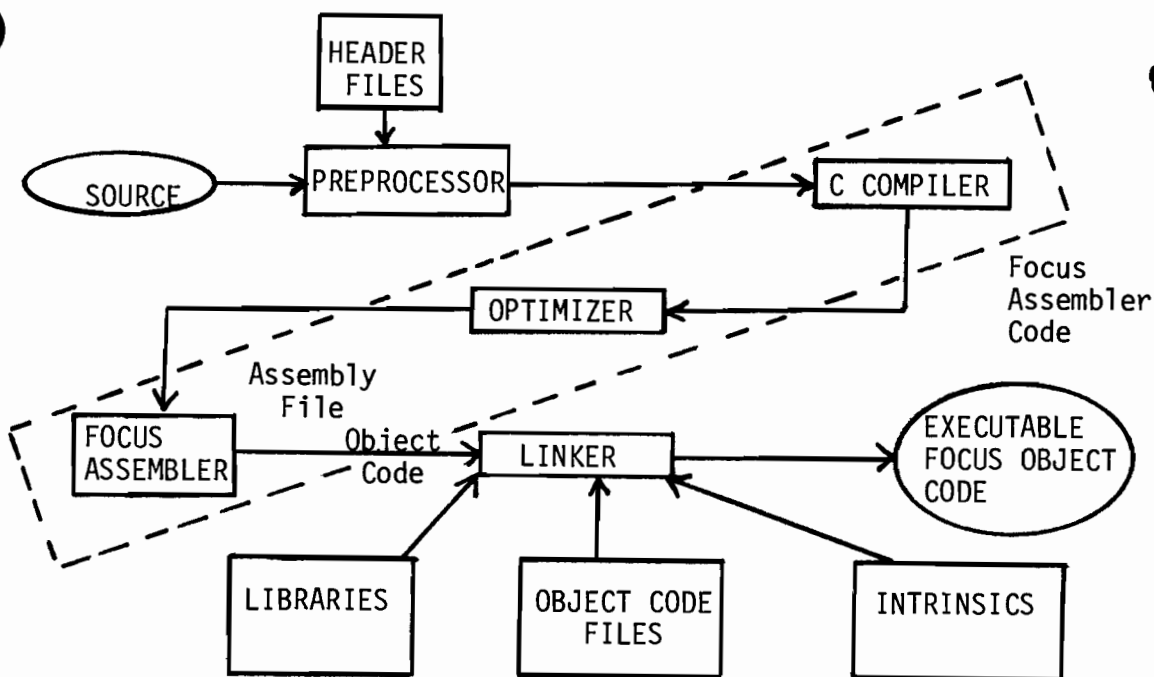
Is "C" A Substitute For Assembly Language?

Under the HP-UX operating system, a "C" compiler will be provided. Depending on what the customer is doing in the high level "C" language, "C" may or may not be a substitute for assembly language.

The bit and stack manipulation features of assembler are present in "C". However, the efficiency of assembler code is not present in "C" code. Also, the I/O instructions, privileged instructions, and absolute addressing capabilities of assembler code are not present in "C".

HP will support the CC command in "C", which has source code as its input and executable object code as its output. The following diagram explains the individual steps in the CC command. The total CC is supported, not the individual steps. The Assembler is simply a part of the Compile statement.

CC Command (Compile "C" Source)



The dashed (---) steps in the diagram will be replaced in the future with a single step, an optimized "C" compiler. No changes will be required to source "C" code to run under the future "C" compiler.

MICROPROGRAMMING AVAILABILITY

No microprogramming capability, such as provided on the HP 1000 with the Writable Control Store, will be available on the HP 9000.

BACKPLANE DOCUMENTATION

There are two backplanes in the HP 9000. The first is the FOCUS Backplane, or the 36MB MPB (Memory Processor Bus.) The second is the I/O Backplane, which connects directly to the IOP (I/O Processor.)

No documentation is available for the FOCUS Backplane. No off-the-shelf-chips will be available for the FOCUS MPB.

The I/O Backplane is accessible through the I/O cards only. These cards include the ASI, MUX, GPIO, HPIB, and PSI (RJE, Ganglia & Custom.) Documentation for these cards is available.

The recommended manner to customize I/O cards is through the Custom PSI (Programmable Serial Interface) card from Roseville. (Availability is March, 1983.) The PSI kit will provide the documentation and the PROM to provide the user the ability to burn his own customized ROM. A breadboard card that allows a user to add his own hardware to an HP "backend" that matches the I/O backplane is being considered as a possible future product.

POWER FAIL STORY

No power fail feature, such as a battery backup, is present in the HP 9000. This is being considered as a possible future enhancement.

RELIABILITY

The Focus chip set was designed with reliability as a major requirement. With every power-up, each major sub-assembly does a complete self-test, followed by the first CPU checking all data paths and sub-assemblies for correct functioning. If all is well, the loader ROM is activated to load the OS and bring up the HP 9000.

If a problem was encountered, the operator is notified by a message on the CRT of the DAWN or by the lighting of LEDs on the CORONA. Extensive protection mechanisms, monitoring temperature in critical locations, is also built into the HP 9000. To protect the system from catastrophic failures due to temperature, the system will notify the operator of the problem and progress through an orderly shut down. Diagnostics will then pinpoint the specific problem.

Should failure occur, the HP 9000 is highly serviceable. Diagnosis, repair, and verification of failures will take less than one hour with simple tools. Less than 5 minutes are needed to diagnose and repair the power supply, memory processor module and the keyboard. Less than 20 minutes are needed to diagnose and repair the printer and the discs.

In summary, the HP 9000 was designed and built with reliability and serviceability from the ground up.

REDUNDANCY

The HP 9000 is not designed as a redundant system. A multi CPU system will not continue to function if one CPU fails.







HP 9000 vs 9845 BASIC

General differences in the BASIC language are mostly syntactical, but there are some things which can cause problems...

- The 9845 allows expressions for bounds in DIM statements. The HP 9000 only allows integer constants. Use ALLOCATE for variable bounds.
- The HP 9000 uses different syntax on some MAT statements than the 9845. Check the specific statement syntax in these cases.
- CRT special function keys are numbered and labeled differently on the HP 9000 and 9845. The HP 9000 has keys 0–7 on the CRT and labels them using ON KEY ... LABEL ...

HP 9000/9845 General BASIC Language Differences

ASSIGN:

The 9000 only allows the form `ASSIGN #num TO filespec [;RETURN<value>]`. The return variable will have a different value than on the 9845. On the 9845, the return variables were a code whereas on the 9000 they will be the number of the error that would have been reported.

AUTO:

This statement has no equivalent on the 9000.

BUFFER #:

This statement has no equivalent on the 9000 since all # statements are buffered.

CAT:

The `msus` is different on the 9000 than on the 9845. For example, if the following statement exists on the 9845...

```
CAT "X:FB"
```

it translates to...

```
CAT ":HP9885,8";SELECT "X" on the 9000.
```

The following statement on the 9845...

```
CAT "X:HB,1"
```

translates to...

```
CAT "HP9895,801";SELECT "X" on the 9000.
```

CAT TO:

All `CAT TO` statements on the 9845 translate to one statement with different options on the 9000. The new format is

```
CAT [source] [TO [destination]] [;options]
```

CHECK READ:

This statement has been changed to `CHECKREAD ON` on the 9000.

CHECK READ OFF:

This statement has been changed to `CHECKREAD OFF` on the 9000.

CREATE

Multiple `CREATE` statements are available on the 9000 depending on which file type is desired. The equivalent to the 9845 `CREATE` statement, which created a `DATA` type file, is `CREATE BCD`. `CREATE` statements also exist to create `DATA`, `ASCII`, `BDAT`, and `PROG` type files.

CRT PRINT PAGE:

The 9845 clears the screen by clearing the buffer. The 9000 scrolls the buffer to clear the screen.

CRT special function keys:

The soft keys on the 9845 are labeled 24-31, but they are labeled 0-7 on the 9000.

DBBACKUP,
DBRECOVER,

4.

HOLE:

These statements have been deleted on the 9000.

**DBCREATE,
DBERASE,
DBPURGE:**

The DBCREATE syntax on the 9000 has been changed to the following.
DBCREATE (BASE\$)[; <optional parameters>].
In addition, the ROOT is not purged in a DBPURGE statement without the SETS\$ parameter. All references to file specifiers must be checked since device numbers are different on the 9000.

DBINFO:

The semantics of this statement have been changed on the 9000.

DEFAULT ON/OFF:

These statements do not work for integer and double data types on the 9000.

DEL:

The 9845 statement...
DEL FN<function name> [TO END]
has been changed to...
DELSUB FN<function name> [TO END] on the 9000.

DEV\$:

The equivalent of this string function of the 9845 is...
SYSTEM\$("MSI") on the 9000.

**DIM,
REAL,
INTEGER,
SHORT:**

These statements on the 9845 allow expressions for bounds whereas the 9000 only allows integer constants in these statements. The ALLOCATE statement on the 9000 allows variable bounds.

**DROUND,
PROUND:**

Since all math on the 9000 is done in binary and all math on the 9845 is done in binary coded decimal, the results of these statements on the two machines may be different.

ERRL:

This function has been changed on the 9000 as follows. The new syntax is...

ERRL(number) or ERRL(Label)
where number/Label specifies the line of interest. The function returns a one if the line number indicated was the line being executed when the last error occurred and a zero if it was not.

**FCREATE,
FPRINT,**

FREAD:

These statements do not exist on the 9000. Their function may be obtained by using ENTER and OUTPUT statements which exist in the I/O option.

File Names:

The following characters are not allowed in 9000 file names...

<
:
/

File Specifiers:

File specifiers on the 9000 differ from those on the 9845. The following statements contain file specifiers and therefore must be modified when transported from the 9845 to the 9000...

GET

INITIALIZE

LOAD/STORE

LOAD/STORE BIN - note: LOAD BIN is not programmable on the 9000

LOAD/STORE KEY - note: The KEY type file does not exist on the 9000 so a new file must be created for use on the 9000

MASS STORAGE IS

SAVE

DBOPEN

VOLUME DEVICES ARE

**FOR/NEXT:**

The 9845 allows both dynamic matching and multiple NEXT statements whereas the 9000 does not allow multiple NEXT statements and does matching statically.

INTEGER Math:

An overflow error may occur on the 9000 which did not occur on the 9845 if the numbers are >32767. In this case, the 9845 treats the numbers as real whereas the 9000 still treats them as 16-bit integers.

IOR:

The IOR statement on the 9845 has been changed to BINOR on the 9000.

KBD%:

This statement has different key codes on the 9000 than exist on the 9845.

LABEL KEY:

This statement does not exist on the 9000. The effect may be obtained by using the ON KEY ... LABEL ... statement on the 9000.

LEX:

This function is not supported on the 9000.

LEXICAL ORDER IS <name>(*):

This statement may be translated to the 9000 if the array is also

6.

translated. Care must be taken with explicit string compares.

LEXICAL ORDER IS STANDARD:

This statement has been changed to...
LEXICAL ORDER IS ASCII on the 9000.

LIN:

The 9845 allows numbers less than or equal to zero as arguments to this function. A zero argument specifies a carriage return and negative arguments specify line feeds. The 9000 does nothing with arguments less than or equal to zero.

LINK:

There is no equivalent to this statement on the 9000. LOADSUB may be used instead.

LOAD ALL:

There is no equivalent to this statement on the 9000.

LOADSUB:

The 9845 has various specific LOADSUB commands. The form,
LOADSUB <filename> [,<startline>][,<increment>] may be translated to
LOADSUB ALL FROM <filename> on the 9000.
In the 9845 form...
LOADSUB <filename> [:<start sub number>[<end sub number>]]
the line number references cannot be translated and must be dropped.
In addition, if a device specifier is used in a LOADSUB statement, it must be changed to be compatible with the 9000.

MAT A=CON,
MAT A=CON(...),
MAT A=ZER,
MAT A=ZER(...):

The first and third statements may be translated directly on the 9000 to

MAT A=(1) and
MAT A=(0).

The second and fourth statements may be translated by using the following statements...

REDIM A(...)
MAT A=(1)

and...

REDIM A(...)
MAT A=(0).

MAT A=IDN(...) may be translated as...

REDIM A(...)
MAT A=IDN.

MAT INPUT:

The 9845 form of this statement,
MAT INPUT A,B,C
may be translated to...

INPUT "",A(*),B(*),C(*) on the 9000.
 If the 9845 form has bounds on an array, for example,
 MAT INPUT A(Number,Number)
 then the array must first be redimensioned on the 9000...
 REDIM A(Number,Number)
 INPUT "",A(*)

MAT READ,
 MAT PRINT,
 MAT READ #,
 MAT PRINT #:

These statements may be translated to the 9000 by dropping the word "MAT" and add a "(*)" to the statement.

MAT SEARCH:

This statement is translatable only under the following conditions.
 If key="(*)" or key="", the condition is MAX or MIN, and the starting address is omitted, the statement may be translated to...

<variable>=<condition>(<name>(*)).

If the above is true but the condition is #LOC the statement may be translated using a large temporary one dimensional array as follows.

MAT SEARCH A(*),#LOC(<rel-op><exp>),(<variable>) is translated to..
 MAT Temp=A<rel-op>(<exp>)
 <variable>=SUM(Temp) on the 9000.

MAT SORT:

If the first character following the array name of this statement on the 9845 is not "(" then MAT SORT may be translated to the 9000 by simply inserting "(*)" immediately following the array name.

MOD:

This operator has been changed to MODULO on the 9000.

MSUS:

The MSUS codes are different between the 9845 and the 9000. Statements which contain an MSUS are...

GET,
 SAVE,
 INITIALIZE,
 LOAD,
 STORE,
 LOAD BIN,
 STORE BIN,
 LOAD KEY,
 STORE KEY,
 MASS STORAGE IS,
 DBOPEN,
 VOLUME DEVICES ARE.

NORMAL:

This statement has been changed to TRACE OFF on the 9000.

ON INT#,
 OFF INT#:

8.

These statements have been changed to ON INTR and OFF INTR on the 9000.

ON KBD [ALL]:

When ON KBD is executed on the 9845, the keys which are not trapped are...

- UP ARROW,
- DOWN ARROW,
- ROLL UP,
- ROLL DOWN,
- TYPEWRITER,
- STOP,
- CONTROL-STOP,
- AUTOST,
- PRINT ALL.

When ALL is specified, the following keys are not trapped on the 9845..

- CONTROL-STOP,
- AUTO STOP,
- PRINT ALL.

When ON KBD is executed on the 9000, the following keys are not trapped...

- PAUSE,
- STOP,
- SHIFT RUN,
- SHIFT PAUSE,
- SHIFT CONT,
- SHIFT STOP,
- ALL,
- NEXT PARTITION,
- SPECIAL FUNCTION KEYS.

When ALL is specified on the 9000, all keys are trapped. In addition, the syntax is different on the 9000.

ON KEY*,

OFF KEY*:

These statements have been changed to ON KEY and OFF KEY on the 9000.

PRINT ALL IS:

The select code for the internal printer has been changed on the 9000. User available select codes are also different on the 9000.

PRINT LABEL:

This statement has been changed to...

PRINT LABEL <label> [TO <device specifier>][RETURN <variable>]
on the 9000. In addition, if a device specifier is used, this must also be changed when transporting code to the 9000.

PRINTER IS:

The select codes for the internal printer and CRT are different on the 9845 and the 9000. Also, HP-IB select codes on the 9000 are only of the form (Select code)*100 + (device number).

PROTECT CODES:

The syntax for specifying protect codes has changed on the 9000 for the following statements.

COPY,
 PROTECT,
 PURGE,
 RE-NAME,
 RE-SAVE,
 RE-STORE,
 ASSIGN.

The new format on the 9000 is...
 PURGE "Filename <password>".

RANDOMIZE:

This statement with an argument on the 9845 is executed by randomizing the fractional part. On the 9000, it is executed by randomizing the integer part.

READ LABEL:

This statement has been changed to...

READ LABEL <label\$>[FROM <device specifier>][:RETURN <variable>]
 on the 9000. In addition, if a device specifier is used, it must be translated to the 9000 format.

REWIND:

There is no equivalent to this statement on the 9000.

ROW (A),

COL (A):

ROW(A) on the 9845 may be translated to...

SIZE(A,RANK(A)-1) on the 9000.

COL(A) on the 9845 may be translated to...

SIZE(A,RANK(A)) on the 9000.

SCRATCH KEY:

This statement has been changed to...

SCRATCH K on the 9000.

SECURE:

This statement does not exist on the 9000 although COMPILE may be used for program security.

STORE ALL:

There is no equivalent to this statement on the 9000.

TABS:

The default tab stops on the 9845 are different from those on the 9000.

TRIG MODE:

The 9000 sets the TRIG MODE to whatever it was in the calling routine upon entering a subroutine whereas the 9845 sets it to RAD. Upon exit, both machines restore the original mode.

TYP:

The TYP(exp) statement on the 9845 has been changed to...

TYP(#ABS(exp),sgn(exp)>0) on the 9000.

10.

TYPEWRITER ON,
TYPEWRITER OFF:

These statements do not exist on the 9000.

[TRACE] WAIT:

These statements have been changed to wait in seconds on the 9000
while the 9845 waits in milliseconds.





HP 9000 vs 9845
BASIC I/O

The area of I/O will probably cause more translation problems than any other area of BASIC.

Major differences are:

- selection of I/O attributes...
The HP 9000 uses ASSIGN to set attributes, the 9845 uses separate statements or the actual I/O statement to set attributes.
- select codes...
The HP 9000 has different default select codes for internal devices than the 9845.
- register I/O...
The HP 9000 has new interfaces and therefore new addressing schemes.

9845 / 9000 I/O ROM Translation Issues

Major changes have been made to the concepts, syntax, and semantics of the BASIC I/O language to result in a cleaner, more consistent definition. For this reason, 9845 I/O ROM statements will not necessarily work correctly, or even syntax properly, on a 9000.

One of the major differences between the 9000 and the 9845 is the use of the ASSIGN statement to set attributes, rather than a separate statement for each attribute. The user must replace his attribute-setting statements with ASSIGN statements which perform the corresponding function. This allows the user to set up all his attributes in one place (the ASSIGN statement) instead of scattering this information all over his program. For example, the 9845 has a NOFORMAT keyword which is inserted in an ENTER or OUTPUT statement to read internal-format variables. The 9000 has a similar capability, but the FORMAT OFF specification can be put into one ASSIGN statement instead of every ENTER/OUTPUT statement.

This method has many advantages, but will cause some problems for current 9845 users. Their programs probably have mode-changing statements throughout the program, and they will have two options. They will either have to group these mode-changers together in ASSIGN statements (the preferred method), or replace each of them with ASSIGNS which have the same result. The example below shows how this could be done: assume the 9845 program contains the statement ENTER D NOFORMAT,X,Y,Z and the new 9000 program has already assigned @Dev to selectcode D with FORMAT ON:

ASSIGN @Dev;FORMAT ON	! Change @Dev to FORMAT OFF
ENTER @Dev;X,Y,Z	! Enter the data
ASSIGN @Dev;FORMAT ON	! Restore to FORMAT ON state

4.

FEATURE-BY-FEATURE TRANSLATIONS

Select code differences

The 9845's select code format allows several formats to specify the same thing. Here is a summary of 9845 formats and their 9000 equivalents:

OUTPUT 7,3,6.0102

The first number is <100 , so it specifies a select code. The following numbers specify HP-IB addresses and subaddresses. The equivalent on a 9000 is 703,7060102. Note the subaddressing, which is after the decimal point on the 9845, is now moved to the left of the decimal point.

OUTPUT 703,706.0102

The first number is ≥ 100 so it specifies a select code and HP-IB address. The only change required here is in the subaddressing: use 703,7060102.

OUTPUT 703.0040

Since the 9845 put HP-IB subaddressing AFTER the decimal point, a special "flag" value is required if you want to have a trailing subaddress of 0. Each subaddress is specified by 2 digits after the decimal point, so this address specifies the subaddresses 0 and 40. 40 is an illegal subaddress (maximum is 31), so the 40 is just a special value to force the 9845 to pay attention to the trailing 0. Since the 9000 specifies its subaddresses on the left side of the decimal point, it doesn't have this problem; the equivalent address is 70300.

OUTPUT -7

This is the same as OUTPUT 7 on the 9000.

OUTPUT -703.0102 or OUTPUT -7,3.0102

This format adds listeners and sends secondary addresses without readdressing the bus. Use SEND to accomplish these goals on the 9000.

OUTPUT/ENTER statements

The OUTPUT and ENTER statements only require translation under the following conditions:

```
OUTPUT sc [type] [NOFORMAT];list
ENTER sc [(type1 | type2 count)] [NOFORMAT];list
type = BINT, WINT, BFHS, WFHS, BDMA, WDMA, WHS
type1 = WHS, BINT, WINT
type2 = BFHS, WFHS, BDMA, WDMA
```

The 9000 device driver selects the best handshake mode, so the various handshake flavors "type", "type1", and "type2" are not translatable. However, the word/byte specification (determined by

the first character of the type field, W or B) can be incorporated into ASSIGN attributes (WORD/BYTE attribute), as can NOFORMAT (FORMAT OFF). The "type2 count" form is NOT directly translatable to 9000 I/O, since there is no good way to specify the "count" parameter. The method used will have to be determined by the individual application.

The BYTE keyword on variable-to-variable ENTER's and OUTPUT's (e.g. OUTPUT A\$ BYTE;list) has no equivalent on the 9000, and is NOT TRANSLATABLE. Likewise, numeric-array sources/destinations in these statements (e.g. OUTPUT A(*)list) is NOT TRANSLATABLE. Only strings and substrings are allowed as the source/destination for variable-to-variable transfers.

SOUTPUT and SENTER should be replaced by OUTPUT and ENTER (with a serial attribute).

WRITEBIN sc;exp [,exp]*
Use OUTPUTBIN sc;exp [,exp]*

EOL sc;seq [,delay]

This statement must be replaced with the ASSIGN attribute EOL. Because of the difference in specifying times, the "delay" would be replaced by "delay/1000". For the "return to default" syntax, EOL sc, use EOL OFF.

The new EOL sequence on the 9000 is used in many more places than the 9845 EOL, which only applied to the L image specifier. The 9000's EOL applies everywhere that the 9845 output a CR/LF (except for the / image spec).

TRL "function"

Use the ASSIGN attribute DELIM. There are two major differences: a TRL terminated strings IN ADDITION TO LF (CHR\$(10)), but the DELIM character on a 9000 REPLACES LF as the terminating character. Also, TRL characters were not stripped from a string, even though LF's still were; however, the DELIM character acts exactly like a LF, so it is stripped.

READBIN(sc) function
Use ENTERBIN(sc).

CONVERT

The CONVERT statement has changed considerably. On the 9845, the syntax was:

```
CONVERT sc ; Mode$ [,Conversion_string$] [, Parity]
CONVERT sc ; Mode$ , Char1$ TO Char2$ [, Parity]
```

6.

Mode\$ specified the direction in which the conversion was to be applied ("I", "O", "OI", or "IO"). Conversion_string\$ gave a table of new (converted) characters, indexed by the character to be mapped. Parity specified the type of parity to be generated (outbound) or checked (inbound): even, odd, zero, one, or none. The second form allowed the user to specify a single character to be mapped into another without specifying an entire table.

On the 9000, the CONVERT and PARITY operations have been changed to ASSIGN attributes, and can be set separately. The new syntaxes for both attributes are:

```
...; CONVERT [ IN | OUT ] BY [ INDEX | PAIRS ] Table$
...; CONVERT [ IN | OUT ] OFF
...; PARITY [ ODD | EVEN | ZERO | ONE | OFF ]
```

Each CONVERT attribute specifies its direction. It is not possible to specify both directions at once; two attributes must be given, one for each direction. The INDEX and PAIRS keywords indicate how the Table\$ will be interpreted: if INDEX, then the incoming/outgoing character is used as an index into the table (just like the Conversion_string\$ on the 9845). If PAIRS, the string is treated as a series of pairs of characters. The first character of each pair is the character to be converted, and the second character is the character it should be converted into. This is similar to the Char1\$ TO Char2\$ form in the 9845.

STATUS sc; var1 [,var2, etc.]

On the 9845, if sc=100, STATUS does a serial poll on an HP-IB interface. Use var1=SPOLL(sc). (Only one variable [var1] is allowed for this form.) If sc<100, STATUS reads device registers. The statement is the same on the 9000, but all the registers are different on the 9000.

RESET

On the 9845, if RESET is executed on anything but an HP-IB interface, it resets the interface by setting bit 5 of R5. On an HP-IB interface, it is equivalent to the CLEAR statement. On the 9000, the interface will be reset regardless of what kind of interface is present. A device-dependent action is performed to do whatever is necessary to reset the interface.

SELECT CODE sc ACTIVE/INACTIVE

There is no direct equivalent for this statement on the 9000. The INACTIVE state can be simulated by ASSIGNing an I/O path-name to the selectcode with the "NULL" driver specified.

ON INT #sc [,prio] GOTO/COSUB/CALL
 OFF INT #sc

Use ON INTR sc... and OFF INTR sc. There are two major differences: the ENTER and OUTPUT statements no longer interrupt when they complete, and device timeouts no longer cause interrupts (see below).

CONTROL MASK sc;mask / CARD ENABLE sc

The 9845 used the CONTROL MASK statement to specify the mask to use for future CARD ENABLE statements. On the 9000, these two statements have been combined into the ENABLE INTR statement. If a mask is specified in the ENABLE INTR statement, the supplied mask is used. If no mask is specified, the previous mask is used, giving semantics similar to the CARD ENABLE statement.

Be aware that the ENABLE INTR statement by itself will NOT write out the interrupt mask! The mask is not actually written out until it is needed; that is, when an ON INTR or WAIT FOR INTR statement is executed.

CONTROL MASK can use a string expression to specify the mask, which is then interpreted as a string of binary digits. However, ENABLE INTR only accepts numeric mask arguments, so use DVAL(Mask\$,2) for the mask. Remember that the 9000 has different interfaces and registers.

Timeout management

The 9845 uses SET TIMEOUT sc;time to set a timeout value for a select code. Then, if a device doesn't respond in this time, an ON INT branch is generated and the TIMEOUT(sc) function in the ON INT ISR is used to determine if this was a timeout "interrupt" or a real interrupt. The 9845 also has SYSTEM TIMEOUT ON/OFF; these have no equivalent in the 9000, since there is no default system timeout.

The 9000 uses a totally different scheme. There is a new ON branch type, ON TIMEOUT, which specifies both the timeout interval and the place to branch to if a timeout happens. This way, the user can have separate handling code for interrupts and timeouts.

COLLECT

This statement helped the user get around a buffering problem on the 9845, but no such problem exists on the 9000. Therefore, no replacement is necessary.

IOFLAG, IOSTATUS, WRITEIO, READIO, WAITREAD, WAITWRITE

Since HP-IO cards do not have registers like the 9845's interfaces, these statements have no equivalent on the 9000.

8.

TOPEN, TCLOSE, TBUF\$, TDISP

There is no direct equivalent on the 9000.

BINAND, BINCMP, BINEOR, BINIOR, BIT, ROTATE, SHIFT

No translation is required. ROTATE and SHIFT are defined to work on 16-bit integers, as on the 9845. However, the 9000's default integer is a 32-bit integer, so DROTATE and DSHIFT should normally be used.

LASTBIT

This function is not supported on the 9000.

OCTAL(dec), DECIMAL(oct)

These functions can be translated as follows:

OCTAL(dec) -> DVAL(DVAL\$(dec,8),10)

DECIMAL(oct) -> DVAL(DVAL\$(oct,10),8)

HP-IB statements

The majority of these statements are the same, although some of them have slightly different semantics. The biggest problem comes from the negative select code addressing scheme (-isc and -scs; see the discussion on select codes, above).

EOI sc;exp

The EOI statement on the 9845 sends out one byte of data with the EOI line asserted. It is frequently used as follows:

OUTPUT sc;Data\$[1;LEN(Data\$)-1]; ! all but last byte

EOI sc;Data\$[LEN(Data\$)] ! last byte w/ EOI This

can be done on the 9000 with OUTPUT sc;Data\$; END.

TRIGGER, CLEAR, REMOTE, LOCAL, LOCAL LOCKOUT, REQUEST,
PPOLL CONFIGURE, PPOLL UNCONFIGURE, PPOLL, PASS CONTROL

No translation is required for these statements. Some statements have slightly different semantics; e.g. REMOTE doesn't drop ATN after raising REN.

The other HP-IB statements require more effort. See the discussions under RESET and STATUS. The ABORTIO sc statement is replaced by ABORT sc.

REQUEST

This statement is compatible with the 9845 with one exception: the 9845 allowed the request value to be specified with a string which was interpreted as a string of binary digits. Use DVAL(Val\$,2) instead.

CONFIGURE, SENDBUS

These statements are replaced by the SEND statement.

CONFIGURE sc TALK=d1 LISTEN=d2 [,d3]*

is replaced by

SEND sc;MTA;UNL;TALK d1;LISTEN d2 [,d3]*.

SENBUS sc;c1a,c1b...;d2a,d2b...;... (etc)

is replaced by

SEND sc;MTA;UNL; CMD c1a,c1b...;DATA d2a,d2b...; (etc)

if sc is <100. If sc>=100, or if it is of the form "sc,addr", or if "sc" contains some subaddressing (e.g. 7,3.0102), it gets a little more complicated, since the SENDBUS then does some bus addressing before sending the user's configuration information. For example,

SENBUS 7,3.0102;1,2;A\$,3,4;5,6,B\$;7,8

is equivalent to

SENBUS 703.0102;1,2;A\$,3,4;5,6,B\$;7,8

and they are replaced by:

OUTPUT 7030102;

SEND 7;CMD 1,2;DATA A\$,3,4;CMD 5,6,B\$;DATA 7,8

PRINT USING/OUTPUT USING/ENTER USING/IMAGE Differences

For the most part, these statements are compatible with the equivalent statement on the 9845. The outbound direction is completely compatible, except that the EOL sequence is output at the end of the statement instead of CR/LF (just like all other output statements). The inbound direction, however, has undergone several changes, which are detailed below.

9845 9000

spec equiv. Semantics

F	K	Freefield numeric input, decimal point radix
N	D	Fixed field numeric input, decimal point radix
InIG	In-11DR	Fixed field numeric input, comma radix (SG = 4DR, etc. 1G = 1D)
T	K	Freefield string
#	+	Terminate statement on EOI only
+	-	Terminate statement on LF only
%	#	Terminate statement as soon as list is satisfied

One major problem arises in the translation of images. Inbound specs #, +, and % are converted to different characters to be more consistent with their equivalents in the outbound direction. However, # and + are also used as outbound image specifiers. An IMAGE statement containing these specs cannot be translated correctly unless you know whether it is referenced by an OUTPUT USING or an ENTER using statement.







HP 9000 vs. 9845 BASIC Graphics

HP 9000 vs 9845 BASIC Graphics

Most programs require only minor changes

Transport issues

Programming Concept Changes: May force substantial revision in some cases

Minor Program Changes : Sometimes needed to achieve 9845-like behavior on 9020

Syntax changes : Easily handled by translator

Unimplemented Statements : Programming work-arounds available

REPORT ON TRANSPORTING
9845 BASIC GRAPHICS STATEMENTS
TO HP9000 Model 20 BASIC

Version 2.0
August 27, 1982

The HP9000 Model 20 BASIC Graphics Option offers significant new graphics capability while providing a high degree of compatibility with 9845 BASIC Graphics. Most 9845 graphics programs will only require minor syntax modifications prior to being executed on the HP9000. Syntax changes to graphics statements may be easily performed by a BASIC translator program. Some 9845 graphics programs may require reprogramming to give acceptable results on the HP9000.

This report describes differences which will occur when syntaxing and executing 9845 graphics statements on the HP 9000 and suggests steps which may be taken to transport 9845 graphics programs. Part I groups transportation actions into four categories: GRAPHICS PROGRAMMING CONCEPT CHANGES, MINOR PROGRAM CHANGES, SYNTAX TRANSLATION, and UNIMPLEMENTED STATEMENTS. Part II lists differences in SYNTAX, SEMANTICS and suggests TRANSLATION steps for each 9845 graphics statement.

Graphics statements of the HP9000 are discussed only as required to transport 9845 graphics programs. Please refer to the FARSTAR ERG or various BASIC manuals for complete discussion of HP9000 BASIC graphics capabilities.

PART I

Actions necessary to transport 9845 BASIC graphics statements to the HP9000 are grouped into four categories.

The first category includes several GRAPHICS PROGRAMMING CONCEPT CHANGES which may force substantial revision to some 9845 graphics programs.

The second category includes MINOR PROGRAM CHANGES which may be necessary to achieve 9845-like behavior on the HP 9000. These changes usually require replacement of a 9845 graphics statement with a short sequence of HP9000 graphics statements. It will be necessary for the programmer to evaluate each change. Thus automatic translation may not be possible.

The third category includes SYNTAX CHANGES which can be made using an automatic translator program.

Finally, there are several UNIMPLEMENTED STATEMENTS.

1. GRAPHICS PROGRAMMING CONCEPT CHANGES

LOGICAL VIEW SURFACE

FARSTAR supports multiple display devices in a device independent manner. User coordinate data is transformed into images represented in logical device coordinates (also known as Graphics Display Units or GDUs). GDUs can be thought of as the dimensions of an internal, "logical view surface". The images created on the logical view surface are mapped onto each enabled physical device. Thus output statements are transformed to a single internal representation and then scaled to fit onto various real output devices.

There is a single "viewing operation" to define how UDU is to be transformed to GDU. It is not possible to establish separate viewing operations for separate devices. Moreover, the viewing operation is unaffected by PLOTTER IS and LIMIT statements.

Thus there is exactly one image and exactly one GDU space. On the 9845, there could be multiple images AND multiple GDU spaces.

The mapping of the logical view surface to devices is normally a best-fit, non-distorting mapping. That is, the logical view surface is mapped to the largest possible rectangular area of the physical device with the same aspect ratio as the logical view surface. It is possible, however, to specify that the full limit area of the physical device be used (using PLOTTER IS or LIMIT). In this case,

the mapping would scale the logical view surface to fit exactly on the device limit area. This full-fit mapping does not, however, alter the range of GDUs for that device. It merely "stretches" the logical view surface to fit the entire limit area.

PROGRAMS THAT RELY ON THE 9845 CAPABILITY TO PRODUCE MULTIPLE, SIMULTANEOUS, INDEPENDENT VIEWS CANNOT BE TRANSPORTED TO THE HP9000. Such programs must be rewritten to output plots sequentially to each plotter.

We recommend the following changes to minimize differences in program behavior. First, begin each program with GINIT. This will terminate all graphics peripherals and reset the viewing specification and all attributes to the default state. Second, add DISTORT to all PLOTTER IS, LIMIT and GRAPHICS INPUT IS statements. The anisotropic GDU mapping resulting from DISTORT may be undesirable in some applications. Finally, reset attributes and viewing specification, as necessary, following PLOTTER IS statements. It may be possible to precede each PLOTTER IS by a GINIT statement to reset attributes and viewing specifications to the default state. GINIT will also terminate all graphics devices, thus this technique cannot be used if multiple graphics devices are being used.

MODAL ATTRIBUTES

Primitive attributes (PEN, LINE TYPE, FILL color, CSIZE, etc) are globally maintained and device independent. It is not possible to set different attributes on different devices. For example, the program sequence below will result in a line drawn using PEN 7 on both plotters:

```
PLOTTER 703 IS OFF
PLOTTER 705 IS ON
PEN 3
PLOTTER 705 IS OFF
PLOTTER 703 IS ON
PEN 7
PLOTTER 705 IS ON
DRAW 100,150
```

The only way to achieve the effect of device dependent attributes is to repeat graphics output to each device after setting attributes desired for that device.

Modal attributes are also unaffected by PLOTTER IS and LIMIT statements. The only way to reset attributes to their default state is using the GINIT statement.

LABEL KEYS

The soft key labelling statements (LABEL KEYS, LABEL KEY #, and KEY LABELS) are not implemented on the HP9000. The ON KEY statement has been extended to include a soft key label.

Programs which use LABEL KEYS but don't require ON conditions may define one or more SCREENS in the soft key label area, then PRINT to these SCREENS.

CURRENT POSITION UPDATES

The current position is maintained in UDU and is not altered when the viewing operation changes. On the 9845 the (device dependent) current position was maintained in device coordinates. Thus the current position may be different after a change to the viewing specification. For example, the value of X,Y after the sequence below will be 1000,2000 NOT the GDU coordinates of the pen position on the plotter.

```
WINDOW -1000,1500,-1000,3750
DRAW 1000,2000
SETGU
WHERE X,Y
```

PLOTTER ARRAY CONTENTS

Output to a plotter array differs in three major ways.

First, the number of array entries and sequence of these entries may not be identical to the 9845. For example, PLOTTER A(*) IS ON will output all current attributes, including area fill color. Programs which access specific rows in these arrays must be changed to use the correct index value.

Second, the coordinates are placed in the array in UDU (after application of the modelling transformation), rather than GDU as in the 9845. Programs which use MAT PLOT to display these arrays must be changed to establish the correct viewing specification.

Finally, the array must be in COMMON. It may be necessary to initialize arrays since COM arrays are not automatically initialized.

2. MINOR PROGRAM CHANGES

Several differences in statement semantics will require minor reprogramming to transport 9845 programs.

DIGITIZE - ON GKEY

The DIGITIZE statement is now completely independent from ON GKEY. It is not possible

to use DIGITIZE (or CURSOR) to return the locator position at the time of the GKEY event. When the event occurs, an event report is saved on an internal event queue. The GKEY statement must be used to read the information saved in the event queue. Thus, the first use of DIGITIZE (or CURSOR) in ON GKEY service routines should be replaced by the GKEY statement.

DIGITIZE - ARROW KEYS

DIGITIZE from ARROW KEYS does not automatically place an echo (tracking cross) on the internal CRT. The echo will only be enabled if TRACK is on for the ARROW KEYS and CRT.

DUMP GRAPHICS

The upper and lower limits of the area to dump are interpreted in GDU. Programs which specify these limits in UDU will require modification to calculate the appropriate GDU value.

Change:

DUMP GRAPHICS Lower,Upper

To:

```
Udu_x=0      ! any x in the 2D window will work
ASKGU Udu_x,Lower;Gdu_x,Gdu_lower
ASKGU Udu_x,Upper;Gdu_x,Gdu_upper
DUMP GRAPHICS ;Gdu_lower,Gdu_upper
```

GSTAT

The GSTAT function is not implemented on the HP9000. The SYSTEM\$ function may be used to obtain graphics system device information.

LORG

The LORG statement now applies justification to the entire character cell. On the 9845 justification was based on the character within the cell. This is no longer possible due the new FONT capability. The following sequence may be used to achieve justification identical to the 9845:

Change:

LORG Org

To:

```
Xjust=((Org-1) DIV 3)/2
IF Org<4 THEN Xjust=Xjust+(1/9)/St_length
IF Org>6 THEN Xjust=Xjust-(2/9)/St_length
Yjust=((Org-1) MOD 3)/2*(8/15)+4/15
LORG Xjust,Yjust
```

Where St_length is the length of the string

To achieve justification that is within a pixel of 9845 x justification, but independent of string

length:

Change:

LORG Org

To:

Xjust=((Org-1) DIV 3)/2)

Yjust=((Org-1) MOD 3)/2)*(8/15)+4/15

LORG Xjust,Yjust

PDIR

On the 9845, the transformed image was rotated by PDIR. On the HP9000, the untransformed object is rotated. Thus, the rotation may introduce distortion not seen on the 9845.

The Angle parameter is interpreted in UDU, not in GDU. The following sequence may be used to get correct rotation angle:

Change

PDIR Angle

To

ASKUU COS(Angle)*100,SIN(Angle)*100;Dxuu,Dyuu

PDIR Dxuu,Dyuu

POLYGON

The POLYGON statement always results in a closed figure. On the 9845 the figure may not have been closed if not FILLED.

The 9845 POLYGON statement (without FILL) should be replaced by the POLYLINE statement. POLYGON with FILL requires no modification.

The 9845 POLYGON always output a regular polygon image, regardless of the UDU coordinate system. On the HP9000, the polygon is regular in UDU space, but may be transformed to an irregular image if the viewing operation is anisotropic.

3. SYNTAX TRANSLATION

A. Several 9845 graphics keywords have been changed in the HP9000. Many of these 9845 keywords are accepted by the HP9000 to provide compatibility for existing programs. In each case the HP9000 keyword will be listed back. These compatibility keywords are listed below:

<u>9845 Keyword</u>	<u>HP 9022 Keyword</u>
SCALE	WINDOW
LOCATE	VIEWPORT
CURSOR	READ LOCATOR
GRAPHICS	GRAPHICS ON
EXIT GRAPHICS	GRAPHICS OFF
ALPHA	ALPHA ON
EXIT ALPHA	ALPHA OFF

B. Several 9845 graphics statements will require minor syntax changes to execute properly on the HP9000. Some of these statements may be easily translated by string substitution. The statements and appropriate translations are given below. Please refer to Part II of this report for more information about semantic changes which motivate these translations.

DUMP GRAPHICS

-Device address
Change
sc,hpib
To
100*sc+hpib

GRAPHICS INPUT IS

-Device address
Change
sc,hpib
To
100*sc+hpib
-Change
"DIGITIZER" or "TABLET"
To
"HPGL"

GRAPHICS INPUT IS ON/OFF

-Device address
Change
sc,hpib
To
100*sc+hpib

LIMIT

-Add ";DISTORT"

MAT AILOT

-Remove MAT
-If (start:end) not present, add (*) to array name

MAT APLOT

-Remove MAT
-If (start:end) not present, add (*) to array name

MAT ARLOT

-Remove MAT
-If (start:end) not present, add (*) to array name

MAT IPLOT

-Remove MAT
-If (start:end) not present, add (*) to array name

MAT PLOT

-Remove MAT
-If (start:end) not present, add (*) to array name

MAT RLOT

```

--Remove MAT
--If (start:end) not present, add (*) to array name

MAT SYMBOL
--Remove MAT
--If (start:end) not present, add (*) to array name

```

```

PLOTTER IS
--Device address
Change
    sc,hpib
To
    100*sc+hpib
--Add ";DISTORT"
--Change
    "9872A"
To
    "HPGL"

```

```

PLOTTER IS ONIOFF
--Device address
Change
    sc,hpib
To
    100*sc+hpib

```

```

POINTER
--Change
    POINTER X,Y,Marker,Color
To
    SET ECHO X,Y;Marker,Color

```

```

POLYGON (without FILL)
--Change
    POLYGON Radius,Num_sides,Sides_to_draw
To
    POLYLINE Radius,Num_sides,Sides_to_draw

```

```

UNCLIP
--Change
    UNCLIP
To
    CLIP OFF

```

4. UNIMPLEMENTED STATEMENTS

The following 9845 graphics statements are not implemented in the HP9000.

```

CONVERGE
DEGAUSS - May be accomplished by:
    ALPHA OFF
    GRAPHICS OFF
GRAPHICS INPUT IS A(*)
GRAPHICS INPUT A(*) IS ONIOFF
GSTAT - Use SYSTEM$

```

KEY LABELS
LABEL KEY# - Use ON KEY LABEL
LABEL KEYS
TRACK A(*) is ON/OFF

PART II

This section lists each 9845 graphics statement and differences in either SYNTAX or SEMANTICS that will be encountered when executing the statement on the HP9000. TRANSLATION steps are included where appropriate.

General differences in error numbers, error messages, math results, etc. are not discussed.

ALPHA

SYNTAX:

- list back as ALPHA ON

AREA COLOR

AREA INTENSITY

AXES

SEMANTICS:

- Current position is updated to intersection.

CLIP

SEMANTICS:

- Does not allow MAX parameters to be greater than MAX parameters.

CONVERGE (not implemented)

Csize

SEMANTICS:

- Character cell height=0 is not allowed
- Slant angle is independent from character cell aspect ratio

CURSOR

SYNTAX:

- list back as READ LOCATOR

SEMANTICS:

- Status string is now device independent. Does not include special data for LIGHT PEN, 9111, etc.
- Cannot be used to read point saved by ON GKEY event. Replace CURSOR in ON GKEY isr with GKEY statement.

DEGAUSS (not implemented)

DIGITIZE

SEMANTICS:

- Status string is now device independent. Does not include special data for LIGHT PEN, 9111, etc.
- Cannot be used to read point saved by ON GKEY event. Replace DIGITIZE in ON GKEY isr with GKEY statement.
- Does not provide automatic tracking or implicit GRAPHICS ON

DRAW

DUMP GRAPHICS

SYNTAX:

- Device address syntax change

SEMANTICS:

- Upper and Lower limits of area to dump are specified in GDU only.
- CRT must be enabled.

EXIT ALPHA

SYNTAX:

- list back as ALPHA OFF

EXIT GRAPHICS

SYNTAX:

- list back as GRAPHICS OFF

FRAME

GCLEAR

GLOAD

SEMANTICS:

- GRAPHICS plotter must be enabled
- Loads only the visible portion of the frame buffer.

GRAPHICS

SYNTAX:

- list back as GRAPHICS ON

GRAPHICS INPUT IS

SYNTAX:

- Device address syntax change
- Device identifier strings "DIGITIZER" and "TABLET" are not allowed

TRANSLATION:

- Change
"DIGITIZER" or "TABLET"
To
"HPGL"

GRAPHICS INPUT IS ONIOFF

SYNTAX:

- Device address syntax change

GRAPHICS INPUT A(*) IS (not implemented)

GRAPHICS INPUT A(*) IS ONIOFF (not implemented)

GRID

SEMANTICS:

- Current position is updated to intersection.

GSTAT (not implemented may use SYSTEM\$)

GSTORE

SEMANTICS:

- GRAPHICS plotter must be enabled

- Stores only the visible portion of the frame buffer.

IPLOT

KEY LABELS (not implemented)

LABEL KEY# (not implemented see ON KEY)

LABEL KEYS (not implemented)

LABEL

LABEL USING

LAXES

SEMANTICS:

- Current position is updated to intersection.

LDIR

LETTER

SEMANTICS:

- Only one string may be entered. That string is output to all enabled plotters.
- Uses current echo type
- Does not do implicit GRAPHICS ON

LGRID

SEMANTICS:

- Current position is updated to intersection.

LIMIT

SEMANTICS:

- Will map Logical View Surface to p1-p2 area in non-distorted, best-fit manner.

TRANSLATION: (if full use of P1-P2 area is desired)

- Add ";DISTORT"

LINE TYPE

SEMANTICS:

- Length of tick marks on line types 9 & 10 is different.
- HPGL repeat length is now correctly implemented

LOCATE

SYNTAX:

- list back as VIEWPORT

SEMANTICS:

- At least one point must lie within limit area.
- VIEWPORT is truncated to Logical View Surface.
- VIEWPORT effect on viewing operation is immediate.

LORG

SEMANTICS:

- Justifies entire character cell

TRANSLATION:

- The following sequence may be used to achieve justification identical to the 9845:

Change:
LORG Org

To:
Xjust=(((Org-1) DIV 3)/2)
IF Org<4 THEN Xjust=Xjust+(1/9)/St_length
IF Org>6 THEN Xjust=Xjust-(2/9)/St_length
Yjust=(((Org-1) MOD 3)/2)*(8/15)+4/15
LORG Xjust,Yjust

Where St_length is the length of the string

To achieve justification that is within a pixel
of 9845 x justification, but independent of string
length:

Change:
LORG Org

To:
Xjust=(((Org-1) DIV 3)/2)
Yjust=(((Org-1) MOD 3)/2)*(8/15)+4/15
LORG Xjust,Yjust

MAT AIPLLOT

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

SEMANTICS:

- Does not update Current Position

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MAT APLOT

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

SEMANTICS:

- Does not update Current Position

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MAT ARPLOT

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

SEMANTICS:

- Does not update Current Position

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MAT IPLLOT

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MAT PLOT

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MAT RPLLOT

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MAT SYMBOL

SYNTAX:

- MAT not allowed
- Array specifier, (*), required

TRANSLATION:

- Remove MAT
- If (start:end) not present, add (*) to array name

MEMORY

SEMANTICS:

- The internal CRT or one of its memory planes must be enabled as a PLOTTER.

MOVE

MSCALE

SEMANTICS:

- The metric scaling will be established on the most recently initialized PLOTTER that is still enabled. Other PLOTTERS and GRAPHICS INPUT devices will have metric scaling only if the hard clip areas on these devices is identical to the device for which metric scaling is established.

ON/OFF GKEY

PDIR

SEMANTICS:

- The angle parameter is interpreted in UDU.

TRANSLATION:

Change
PDIR Angle
To
ASKUU COS(Angle),SIN(Angle);Dxuu,Dyuu
PDIR Dxuu,Dyuu

PEN

SEMANTICS:

- PEN action is independent of background color.

PENUP

PLOT

PLOTTER IS

SYNTAX:

- Device address syntax change
- Device identifier strings "9872A" and "INCREMENTAL" are not allowed.

SEMANTICS:

- This statement has no effect on values for primitive attributes (PEN, LINE TYPE, AREA COLOR, etc) or viewing parameters (WINDOW, VIEWPORT, etc)
- The range of GDU coordinates in x and y directions is fixed by the aspect ratio specified in GINIT. Thus GDU space is totally device independent.
- The Logical View Surface is mapped to the P1-P2 area of the plotter in a non-distorting, best-fit manner.

TRANSLATION:

- Device address (sc,hpib -> 100*sc+hpib)
- Add ";DISTORT"(if full use of P1-P2 area is desired)
- Change
"9872A"
To
"HPGL"

PLOTTER IS ONIOFF

SYNTAX

- Device address syntax change

TRANSLATION:

- Device address (sc,hpib -> 100*sc+hpib)
- Add ":DISTORT"

PLOTTER IS A(*)

SEMANTICS:

- Output to array will occur in a different sequence, and additional output may occur. Programs which directly access these arrays by row index may require some reprogramming to establish new index values.
- Array must be in COM

PLOTTER A(*) IS ONIOFF

POINTER (not implemented)

SEMANTICS:

- The POINTER statement was used to establish the position of both the input device locator and the echo on an output device. SET ECHO is now used to set the echo position and type. Set LOCATOR is used to set the input device locator position.
- Marker types 4,5,6 do not result in rubberband lines

-Marker type 0 removes echo but does not disable movement

TRANSLATION:

Convert

POINTER X,Y,Marker,Color

To

SET ECHO X,Y,Marker,Color

POLYGON

SEMANTICS:

- The POLYGON statement now always results in a closed polygon whether it is filled or not. POLYLINE may be used to create an unfilled, polygon that is not closed.
- POLYGON output will be irregular if viewing operation is anisotropic.

TRANSLATION:

All instances of POLYGON without FILL should be translated as follows:

Change

POLYGON Radius,Num_sides,Sides_to_draw

To

POLYLINE Radius,Num_sides,Sides_to_draw

RATIO

SEMANTICS:

- Returns aspect ratio of the Logical View Surface, and is device independent.

RECTANGLE

RPLOT

SCALE

SYNTAX:

- list back as WINDOW

SETGU

SETUU

SHOW

TRACK IS ONIOFF

TRACK A(*) IS ONIOFF (not implemented)

UNCLIP (not implemented)

TRANSLATION:

- convert UNCLIP to CLIP OFF

WHERE

SEMANTICS:

The current position is maintained in user units. The CP coordinate values are not changed when the viewing transformation changes. Thus the sequence

SETGU

WHERE X,Y

will return values set by the most recent graphics

output statement. These values will not be the GDU coordinates of the plotter pen position.

To obtain the GDU coordinates of the CP execute the following sequence:

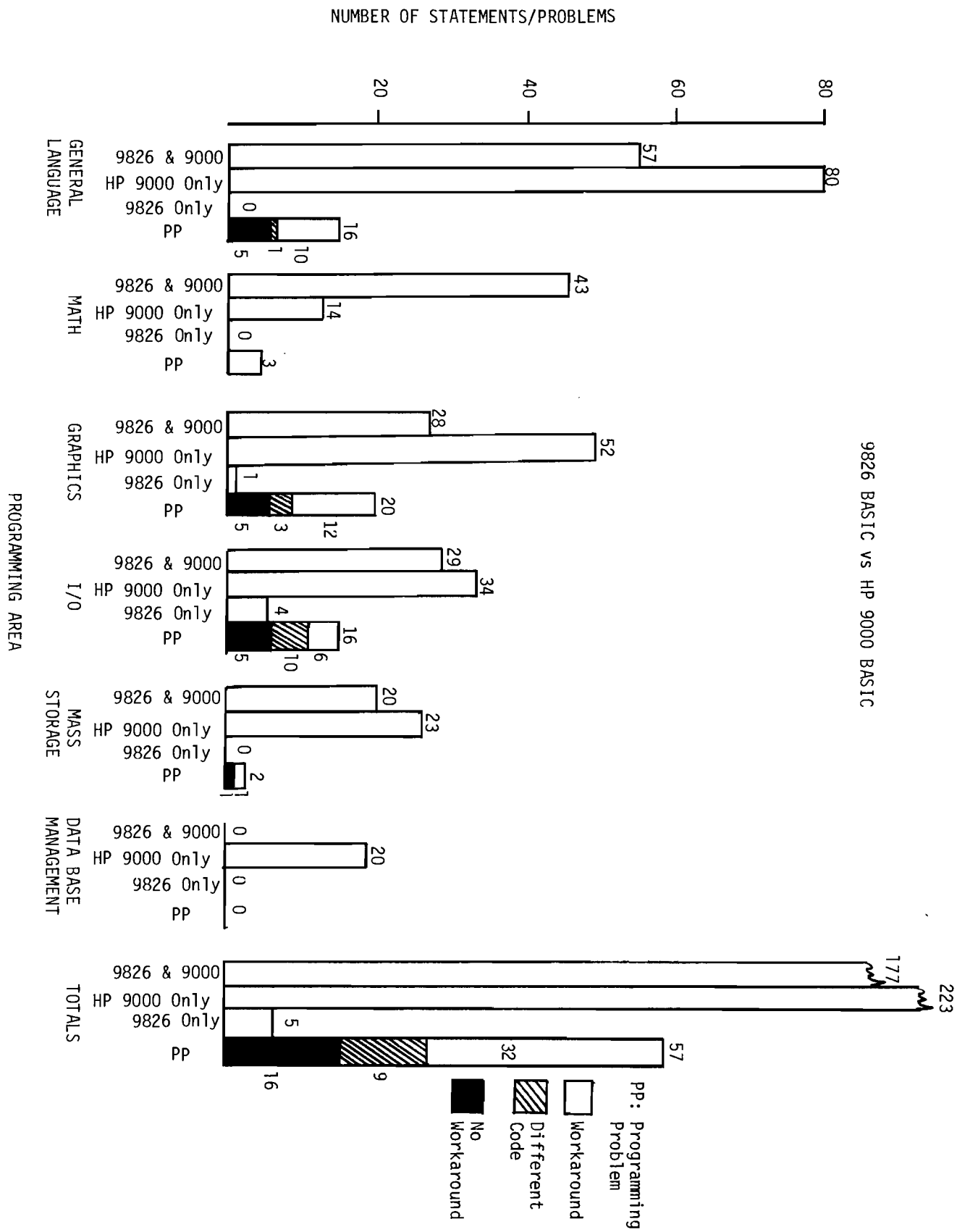
```
WHERE X,Y
ASKGU X,Y;Gdux,Gduy
```







9826 BASIC vs HP 9000 BASIC



9826 BASIC vs HP 9000 BASIC

Programming problems are divided into the following categories:

Human Interface	Graphics
General Language	I/O
Math	Mass Storage

Each problem is discussed below, along with its subcategory and its workaround where appropriate. Subcategories are indicated by W (Work-around), DC (Different Code for Chipmunk and Dawn), and NW (No Work-around).

Programming Problems.

Human Interface

- W This area covers differences like CRT size, location and number of labels for special function keys, and position of editing and program control keys. Differences are mainly ones of convenience and do not actually affect program execution. The only difference which could affect execution is the fact that Chipmunk can have keys 0-9 labeled and Dawn can only label keys 0-7. This program is easily fixed, or can be taken into account when a program is initially written.

General Language

- W 1) The 9826 and HP 9000 re-allocate and re-initialize Common under different conditions. All problems generated by this difference can be solved if the COM statement appears in each main program and if all variables that are in Common are initialized before use.
- W 2) Chipmunk can label two more special function keys than Dawn.
- W 3) Chipmunk stores whole numbers without a decimal point as real numbers if they are >32767. Programs could generate OVERFLOW on Dawn since it will store these numbers in a DOUBLE format. The workaround is to add a decimal point.
- W 4) Chipmunk and Dawn determine dimensionality of implicitly dimensioned arrays at different times. Since arrays should be dimensioned before use, the workaround is to program correctly in the first place.

4.

- W 5) Chipmunk and Dawn evaluate loop values in different order. This can cause problems if side effects are involved in evaluating the values. Programs which rely on side effects to run correctly deserve to get caught when run on different machines.
- W 6) Chipmunk and Dawn system messages may be different. Since most programs will use the error number, this shouldn't be a problem.
- W 7) Chipmunk uses deferred binding in expression evaluation and Dawn uses immediate binding. This is a problem only when side effects are involved.
- W 8) Chipmunk and Dawn BIN, PROG, and compiled subprogram files aren't compatible. LIF has been provided to transport programs so the workaround is to save source code in LIF files and re-compile as necessary.
- DC 9) Since the CRT's are different sizes, line feeds occur in different places when ENTER is done from the CRT. This problem also arises between the 9826 and 9836. Different code (chosen through SYSTEM\$) can be used for a workaround.
- NW 10) Branching on OVERFLOW might behave differently on Chipmunk and Dawn since Dawn has a DOUBLE format. This difference must be taken into account when initially writing a program to run on both machines. The lab feels this problem may never occur.
- NW *11) If an error occurs while processing a DELSUB. . . TO END, Chipmunk and Dawn may have deleted different subprograms. Chipmunk starts at the end and deletes to the named sub and Dawn deletes beginning from the sub to the end. A problem could occur if ON ERROR CALL was used and Chipmunk had already deleted the sub.
- NW *12) If a null string is passed by value to a sub in Chipmunk, it can be assigned a single character. This is a bug which will remain.
- W *13) Chipmunk allows ON <event> CALL X where X is a sub with all optional parameters and Dawn requires that the sub have no parameters. Chipmunk will document the Dawn definition so the workaround is not to use this Chipmunk feature.
- NW *14) If a REDIM has changed the dimensions of an array in COM, Chipmunk and Dawn may match COM differently at prerun. This is a Chipmunk bug and will be changed in the future.

NW *15) W, L, and - IMAGE specifiers have different affects on Chipmunk and Dawn.

W *16) Chipmunk allows MAT SORT A, but Dawn requires MAT SORT A(*). The workaround is to use A(*) in Chipmunk programs.

MATH

W 1) Dawn conversion routines used for decimal string to binary and from binary to decimal string are implemented at the FOCUS level using routines which are very fast. They do not conform to the proposed IEEE Math Standard Implementation Guide, nor to Chipmunk routines. Software routines conforming to the IEEE standard were tested against the hardware routines and provided the following results.

The hardware routine for decimal to binary conversion generates errors of up to 3 binary counts 42.8% of the time for worst case numbers near powers of 10. For best case numbers near powers of 2, errors mean that any bits due to carries in the internal representation of the number could be wrong.

Hardware binary to decimal conversion generates <16 good digits 32.9% of the time (again worst case for powers of 10), and 22.5% of the time for best case powers of 2. It is interesting to note that when numbers were converted from binary to decimal and back to binary, only 51.6% came back correct for worst case powers for 10 and 55.2% for best case random.

An even more interesting test is that for decimal to binary and back to decimal using the hardware routines vs using the software routines. In this test, <16 good digits were generated 67% of the time using the hardware routines. In the worst case, only 14 digits were good (8% of the time). The software routines returned the same string as the original all of the time. For example:

original string:	9.999 999 999 999 815 2 E-308
hardware routine	
returns:	9.999 999 999 999 813 3 E-308
strings must be	
rounded to 14	
digits to match:	9.999 999 999 999 8 E-308

This problem could be substantial if high precision numbers are being passed around the EPN between Chipmunks and Dawns as ASCII strings. The workaround for conversion problems is, of course, to throw away precision. (We might note that the software routines tested are the same as those used on Chipmunk, except that Dawn's are good internally to 19 digits and Chipmunk's are good to 16 digits.)

2) Both Chipmunk and Dawn use the transcendental routines proposed by Cody and Waite, sort of, as follows:

Agreement with Cody Chipmunk Dawn		Deviation from Cody Chipmunk Dawn	Chipmunk/Dawn Differences
SIN	Experimental*		---
COS	Experimental		---
	ASN		---
	ACS		---
TAN	TAN		---
ATN	ATN		---
	LOG		---
	LGT		---
X^Y		X^Y; Y<=8	?
SQR	SQR		---
EXP	EXP		---

*experimental routines using Cody were also tested.

All of these differences will only be seen in the least significant bits of the internal representation of the number. They may also be seen in the least significant digit of the decimal string of the number. (Assuming the conversion introduces no additional error.) The real problem here is that nobody conforms to anything recognized by the industry. If we're going to say we conform to the Cody and Waite routines, then some changes have to be made. Again, the workaround for this problem is to use less accuracy. (Jack Cooley tells me that we plan to be compatible with Chipmunk whether or not they conform to Cody. The only transcendental which might be different is the X^Y for $Y \leq 8$, which gives extra accuracy and performance to Dawn.)

W 3) Since Cody doesn't specify a DEG mode, Chipmunk and Dawn may differ in this mode. The workaround is to use less precision.

GRAPHICS

W 1) Several system parameter defaults are different on Chipmunk and Dawn due to different CRT sizes. The workaround, and a better programming practice anyway, is to set the parameters explicitly as variables at the beginning of the program.



- W 2) In Chipmunk, PEN 0 complements each pixel, whereas on Dawn it's a NOP. This means that extra code is needed for a Chipmunk program using this feature to run on Dawn. The best thing is not to use this feature from the very beginning when writing a Chipmunk program which will run on Dawn. (If a program is doing rubber banding on Chipmunk, it may have to use this feature since the software/hardware isn't as full as that on Dawn. In this case, extensive rewriting may be necessary.)
- W 3) Dawn doesn't have a DUMP DEVICE IS statement since it doesn't have a DUMP GRAPHICS key. The fix is to add a device number to any DUMP GRAPHICS statements in the program.
- W 4) Chipmunk and Dawn have different default aspect ratios so the RATIO statement returns different numbers on the two machines. Since the aspect ratio can be changed in Dawn's GINIT statement, this problem goes away.
- W 5) Chipmunk allows multiline function calls in LABEL statements and Dawn doesn't. The workaround is not to use this feature.
- W 6) Chipmunk and Dawn map the viewport onto graphical input devices differently. Dawn initializes all active devices to the same aspect ratio and Chipmunk maps the viewport directly onto the physical limits of each device. This problem could lead to input and output devices having different viewports and therefore different scaling when a program is run on Dawn, all the user must do is programmatically set P2 on the input device so that the aspect ratio is the same for graphical input and output devices.
- W 7) If the parameter used in a PFN statement is <0, Chipmunk uses line type 0 and Dawn uses the current line type. Also the default repeat length of the line type is 5% on Chipmunk and 4% on Dawn. The workaround is to set the desired line type and repeat length explicitly.
- W 8) The PLOTTER IS statement on Chipmunk deactivates the previous plotter whereas on Dawn it adds the new plotter to the list of active plotters. A plotter can only be deactivated on Dawn using the PLOTTER . . . IS TERMINATED statement.
- NW *9) In Chipmunk, VIEWPORT only affects the subsequent WINDOW or SCALE to VIEWPORT mapping whereas in Dawn the affect is immediate. Chipmunk intends to adopt the Dawn definition in a later release.
- W *10) CLIP OFF is the default for Chipmunk and CLIP ON is the default for Dawn. The workaround is to explicitly state CLIP OFF in Chipmunk programs.

8.

- W *11) Chipmunk and Dawn differ as to when the CRT is the active plotter. The workaround is to use Moonunit which is compatible with Dawn.
- W *12) On Dawn, GINIT turns graphics off and on Chipmunk it only initializes graphics. Dawn may change to the Chipmunk definition. In the meantime, the workaround is to turn graphics on after a GINIT.
- DC *13) On Dawn PLOTTER IS does a GCLEAR and on Chipmunk it does not. Chipmunk will adopt the Dawn definition.
- NW *14) If a positive nonexistent pen is specified on Chipmunk, no range check is done. On Dawn, it is mapped into an allowable subset of pen numbers. Chipmunk will adopt the Dawn definition.
- W *15) Chipmunk does a paper advance with GCLEAR to an HPGL device. Dawn requires an optional parameter to cause the advance. Chipmunk will adopt the Dawn definition. In the meantime, the parameter must be added to a Chipmunk program.
- NW *16) Dawn does block justification of multiple line LABEL's and Chipmunk does not.
- NW *17) In Chipmunk, tickmarks associated with GRID are not visible outside the window and in Dawn they are.
- NW *18) The RPLLOT local origin is not affected by PIVOT on Chipmunk, but it is on Dawn.
- DC *19) Line type number 4 is different on Dawn and Chipmunk.
- DC *20) The default cursor types used by SET ECHO are different on Dawn and Chipmunk. The workaround is to explicitly set the desired type in the Dawn version of the program.

I/O

- W 1) Chipmunk leaves I/O path names valid in the Common area at STOP, END, LOAD, GET, and SCRATCH, whereas Dawn invalidates them at these times. This means that a Chipmunk program using main program overlays with I/O path names in Common won't work on Dawn. The workaround for this problem is to use subprogram overlays since both machines treat LOADSUB the same way leaving I/O path names in Common open. Since modular programming is becoming popular, this is the method of choice anyway.

- W 2) The order of servicing all ON<event>'s with the same priority may be different. This can be avoided by using different priorities.
- W 3) Dawn doesn't have a RESET option on SUSPEND INTERACTIVE since there isn't a RESET key. This is a low probability problem and occurs due to hardware differences. Apparently, a hard reset on Dawn will generally require a power-on sequence to gain control, so it was left off entirely.
- DC 4) Chipmunk and Dawn have different select codes. This affects not only I/O, but also 9826 peek & poke and powerfail since these are accessed through select codes. A program must have different code for the two machines and select the appropriate sections (through SYSTEM\$) in order to run correctly. A program currently running on Chipmunk can best be transported if @ names are used for select codes.
- DC 5) Chipmunk and Dawn will access I/O card registers differently. Again, a program must execute different code depending on which machine is running it.
- NW 6) When data is entered from the keyboard, Chipmunk discards the unread contents of the keyboard buffer and Dawn doesn't. I'm told that Dawn's implementation may change since the current one tends to leave garbage carriage-returns and line-feeds in the keyboard buffer.
- NW 7) Chipmunk and Dawn can give different results when a program uses ENTER USING "nD". Both differ from the 9845 implementation also. On the 9845, n characters are read, non-numeric characters thrown away, and a number built. On Dawn, n characters are read, non-numeric characters thrown away, and the first number found is taken. On Chipmunk, n characters are read and the first number found is taken. For example: if the string is "-1-2-3", the 9845 gets 123 as its number and both Dawn and Chipmunk get -1. However, if the string is "-ddd.1", the 9845 gets -.1, as does Dawn. But Chipmunk gets .1. In this case, depending on the data entered, Chipmunk and Dawn could generate different results. (Note that the DCD BASIC Committee recommends that this difference be approved as is.)
- DC *8) In Chipmunk, the W image specifier describes both the data and the way it is sent to the interface. With Dawn, only the data is described by the image specifier and the handshake width is described by the I/O attributes WORD and BYTE. Chipmunk does not support these attributes, and defaults to BYTE handshake. Chipmunk will support these attributes in the future.

10.

- W *9) When an OUTPUT is performed to the keyboard, Dawn discards all keys after the first closure key and Chipmunk does not discard them. However, Chipmunk discourages use of more than one closure key because of unpredictable behavior. The workaround is not to use more than one closure key.
- NW *10) If an OUTPUT USING statement only outputs an END, Chipmunk suppresses the EOL sequence and does not generate an EOI an HPIB interface. Dawn sends both EOL and EOI. This is a Chipmunk bug and will be fixed in 3.0.
- W *11) Chipmunk allows function calls in INPUT lists, and Dawn does not. The workaround is not to use this feature of Chipmunk.
- NW *12) In Dawn, if an ON condition is logged while a subexit is in progress, if the priority is appropriate, the subexit is aborted and the ON is serviced, then the subexit is restarted. In Chipmunk, some such ON's may be lost, and some may be logged in the new context.
- DC *13) Dawn does not support fast handshake mode for TRANSFER's and Chipmunk will.
- DC *14) Dawn does not have PPOLL RESPONSE since the HPIB card does not support this feature.
- NW *15) Chipmunk cannot detect a timeout during overlapped TRANSFER's, but Dawn can. Also, in Chipmunk timeouts become longer when an overlapped operation is in progress.
- W *16) Chipmunk applies WORD and BYTE attributes to devices, whereas Dawn applies them to files and buffers as well. A workaround may be necessary using ENTERBIN and OUTPUTBIN.

MASS STORAGE

- W 1) Chipmunk has a media backup statement and Dawn doesn't. Since Dawn will have a utility to do media backup this doesn't present a serious problem.
- NW 2) Chipmunk can catalog BIN files and Dawn can't. Chipmunk uses the same file format for BIN files as for PROG files so this feature was free. Since Dawn has different file formats for these two types of files, it can't catalog a BIN file. (Jack Cooley is pushing to get Chipmunk to drop this one, perhaps by not documenting it.)



FORTRAN/9000

FORTRAN/9000

- * ANSI FORTRAN 77
- * Includes MIL—Standard—1753 Extensions
- * Supports recursion
- * Allows calls to other HP—UX languages
- * No FORTRAN 66 compile option
- * Re—entrant
- * Standard FORTRAN 77 compiler directive

HP 9000 FORTRAN 77 WHITE PAPER

The following paper was developed to provide you with compatibility information about the HP 9000 FORTRAN 77 compiler. This paper discusses the following:

- * FORTRAN 77 Overview
- * FORTRAN 77 Comparison Table
- * Detailed Comparison between FORTRAN 77/9000 and FORTRAN 77/4000

FORTRAN 77 OVERVIEW

High level languages on the HP 9000 are an important part of the total HP 9000 offering. Today, FORTRAN dominates the scientific computing world and will continue to do so in the foreseeable future.

The FORTRAN 77/9000 compiler is a full implementation of the ANSI X3.9 - 1978 Standard for FORTRAN. Many extensions have been added to provide a more structured approach to program development and more flexibility in computing for scientific applications. The table below summarizes the major features of HP and DEC FORTRAN offerings. Following is a detailed comparison between FORTRAN 77/9000 and FORTRAN 77/1000.

FORTRAN COMPARISON TABLE

Features	ANSI 77	HP 7X (1000)	HP 77 (9000)	DEC 77 (VAX)	DEC IV+ PDP-11
ANSI 77 (Full)	Y	Y	Y	Y	N
ANSI 66	N	Y	N	Y	Y
Global Optimizer	N	N	N	Y	Y
Symbolic Debugger	N	Y	N	Y	
Reentrant Code	N	N	Y	Y	Y
Recursion	N	N	Y		
Conditional Compilation	N	Y	Y	Y	N
Range Checking	N	N	Y		
BYTE Data Type	N	N	N	Y	Y
CHARACTER Data Type	Y	Y	Y	Y	Y
Hollerith Data Type	N	Y	Y	Y	Y
Double COMPLEX	N	Y	Y	Y	N
IF-THEN-ELSE	Y	Y	Y	Y	N
DO WHILE	N	Y	Y	Y	N
PARAMETER	Y	Y	Y	Y	Y
IMPLICIT NONE	N	Y	Y	N	N
OPEN,CLOSE	Y	Y	Y	Y	Y
INQUIRE	Y	Y	Y	Y	N
ENCODE,DECODE	N	Y	N	Y	Y
Alternate Returns	Y	Y	Y	Y	Y
DO w/ Extended Range	N	Y	N	Y	Y
Debug Lines	N	Y	N	Y	Y
Upper/Lower Case	N	Y	Y	Y	N
Long Identifiers	N	Y	Y		
Unlimited Dimensions	N	N	Y		
IBM direct access I/O	N	Y	N	Y	Y
Price		4,500	6,000	7,000	5,800

DETAILED COMPARISON BETWEEN FORTRAN 77/9000 AND FORTRAN 77/1000

1. One of the major differences between FORTRAN 77/1000 and the 9000 is that the 1000 FORTRAN 77 has the capability of running in ANSI 66 mode. This was not included in FORTRAN 77/9000.

One of the ANSI 66 and 77 conflicts which can be resolved in FORTRAN 77/9000 is the 66 style one-trip DO loop (by using the \$onetrip directive). Other ANSI 66 and 77 conflicts which could be resolved in FTN7X (by specifying 66 mode) but not in FORTRAN 77/9000 are:

- * computed GOTO value out-of-bounds condition
- * intrinsics declared in EXTERNAL and type statements
(use the INTRINSIC statement instead in FORTRAN 77/9000)

Refer to Chapter 8 of the HP 1000 FORTRAN 77 Reference Manual for a description of each of these conflicts.

2. In FORTRAN 77/1000 the range of the DO loop can be extended by passing control out of the loop and then back into it. This is not included in the 9000 FORTRAN compiler.
3. The EMA Statement has been removed from the 9000 version. Programs don't need this feature, since they already have a large address space.
4. In FORTRAN 77/1000 a multidimensional array can be referenced by a single dimension in the EQUIVALENCE statement (pg. 3-56 of the HP 1000 FORTRAN 77 Reference Manual). This extension was not implemented in the FORTRAN 77/9000.
5. On the 1000, the FUNCTION, SUBROUTINE, BLOCK DATA, and PROGRAM statements allow comments to be specified. This is not allowed on the 9000 (since there is no NAM record).
6. Instead of the program type and priority being specified in the PROGRAM statement, formal parameters can be specified and can be passed values from the run string.
7. On the 9000 there is no LIST or NOLIST option for the INCLUDE Statement or Directive as there is on the 1000 (pg. 3-78 of the HP 1000 FORTRAN 77 Reference Manual).

8. There is no CALL EXIT extension implemented on the 9000.
9. On the 9000 variables can be initialized in the type statement. This is not allowed on the 1000.
10. The \$FILES directive set aside room for the DCB on the 1000. This directive is not needed on the 9000 since HP-UX files don't have DCBs.
11. Recursion is allowed on the 9000, but not the 1000.
12. The \$ALIAS directive on the 9000 does not include the NOABORT, DIRECT, NOEMA options.
13. The OPTION, ONETRIP and the RANGE (for run-time subscript range checking) directives are new for the 9000.
14. In FORTRAN 77/1000 true = any negative value and false = any non-negative value. In FORTRAN 77/9000 true = any non-zero value and false = 0 (thus making FORTRAN 77/9000 compatible with C).
15. The intrinsics (as documented in Appendix B of the appropriate FORTRAN 77 Reference Manual) are the same for the 9000, except for the following:

PCOUNT, ISSW, EXEC, REIO
16. On the 9000 32-bit integer values can be inputted or outputted in octal format. This was not allowed on the 1000.
17. There is no FTN control statement in the FORTRAN 77/9000 language.
18. Compiler invocation is different and there are different compiler options and directives.
19. In FORTRAN 77/9000 the Pascal list format will be used. This means that a '+' will not appear in the list file to indicate the lines that have been included using the INCLUDE Statement or directive.
20. The carriage control character '*' (no advance) is not implemented on the 9000. A '+' is included instead for no advance (ANSI 77 standard).

21. As an extension to the ANSI 77 standard, FORTRAN 77/9000 permits mixed character and integer entries, as well as mixed lengths among character entries. On the 1000 if an entry name was of type character, each entry name and the function name must also be of type character, and all such entities must have a length specification of the same integer value.
22. On the HP 1000, if the FORMAT statement specifies a record size > 67 words, LGBUF must be called. On the 9000 this restriction is not true, and nothing special needs to be done.
23. On the 1000, the sizes w,d and n fields within the format specifications are checked at compile-time for a value > 2047. On the 9000 these fields are not checked at all at compile time.
24. FORTRAN 77/9000 allows unlimited number of array dimensions, whereas FORTRAN 77/1000 only allows 7 dimensions.
25. Six byte REAL data type is not implemented on the HP 9000, but it is on the HP 1000.
26. On the HP 1000, as an extension to the ANSI 77 standard, continuation lines were unlimited. On the HP 9000 only 19 continuation lines are allowed.
27. FORTRAN 77/1000 included compatibility features that are not a part of the ANSI 77 standard. The compatibility features that are included in FORTRAN 77/1000 and are not included in FORTRAN 77/9000 are:
 - * extended precision type
 - * I and J suffixes to constants
 - * integer variable name where character expression required in an input/output statement
 - * DECODE and ENCODE statements
 - * improper array dimensioning in EQUIVALENCE statement
 - * record number connected to unit number (old-style direct access I/O)
 - * statement function in EXTERNAL statements as arguments
 - * two-way arithmetic IF statements
 - * parentheses around simple I/O lists
 - * D in column 1 denoting debug lines
 - * \$ as statement separator
 - * complex variables, slash(/) and EOLN in list-directed input
 - * mixed-type operations on complex and double precision operands

- * hollerith constants are stored as a different type depending on the number of characters and on the mode (66 or 77).
- * unformatted I/O and paper tape length words

Refer to Chapter 8 of the HP 1000 FORTRAN 77 Reference Manual for a detailed description of each of these features.

28. The ASSIGN and the Assigned GOTO Statements require a 32-bit integer variable on the HP 9000 (since addresses are 32-bits). This was not a restriction on the HP 1000.



PASCAL/9000

2.

PASCAL/9000

- * Implementation identical to Pascal 1000
- * Allows calls to other HP-UX languages
- * Generates HP 9000 native code (not P code)
- * Compiler directive for proposed ANSI standard PASCAL

HP PASCAL/9000 WHITE PAPER

To expand outside the traditional Desk-top computer market, high-level languages, such as Pascal and FORTRAN, are needed on the HP 9000. This paper provides a brief description of Pascal/9000 and its compatibility with other HP Pascal compilers. A chart of comparable Pascal compilers from other vendors is also given.

Pascal is rapidly gaining acceptance in industry due largely to its wide academic acceptance. Standard Pascal features include a versatile yet concise set of program control constructs, strong typing, and block-structuring. Block structuring allows for easy debugging and reading, and facilitates documentation, modification, and maintenance. HP Pascal compilers are based on the draft proposed ANSI standard.

HP Pascal Compatibility Overview

The following is a rough outline of the Pascal compatibility picture as it exists today within HP (9/1/82). The following are covered:

1. HP Standard Pascal differences from the ISO/ANSI standards.
2. Actual implementation differences from HP Standard Pascal.
3. List of compiler options supported by each implementation.

These lists are purposely brief, and are subject to change with future software updates. For more detailed information, the appropriate language reference manuals, the HP Standard Pascal document, and the ISO Standard Pascal documents should be consulted.

1. HP Standard Pascal Differences from ISO/ANSI standards

HP Standard Pascal allows:

General:

- . ASCII character set
- . Compiler options of the form \$<option>, <option>, ... \$
- . Underscore character ('_') allowed in identifiers
- . Identifiers may be up to a source line in length
- . Predefined constants maxint and minit define the largest and smallest integer respectively
- . Predefined functions hex, octal, binary (convert a string, or PAC (Packed Array of Characters) containing digits in base 16, 8, and 2 respectively, to an integer)

Compilation units:

- . Allows a collection of routines with no main program body to be compiled separately from the main program
- . Allows modules to be defined to facilitate program structuring

Routines:

- . Allows FORWARD and EXTERNAL procedures and functions
- . Functions can return structured types

Declarations:

- . CONST, TYPE, and VAR sections may appear in any order, multiple occurrences per block
- . Constant expressions allowed in CONST section
- . Constant expressions allowed in subrange definitions
- . Allows structured constants (constants of type array, record, set, string and PAC). ISO/ANSI allows a limited form of set and PAC constants

Types:

- . Predefined type LONGREAL
- . Strings
 - Syntax for non-printing characters in string literals
 - String literals padded with blanks in assignments and comparisons
 - Can access a component character of a constant string
 - Can assign, compare, pass strings with different current and max lengths
 - Conversions between strings and Packed Arrays of Chars
- Predefined procedures: strappened, streadd, strwrite, setstrlen, strmove, strinsert, strdelete
- Predefined functions: strlen, strmax, strpos, strttrim, strntrim, strprt, str

Feature products

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
ARCHIMEDES	No	Yes	No	No
Async Terminal Emulator	97076A	97056A	No	No
CRAYONS	No	Yes	No	No
DATAcap	No	No	92080A	No
HP-DSN	Phase3	No	91750A	91750A
EGS (Pigpen)	No	No	No	No
GANGLIA	Yes	Yes	Yes	Yes
GRAPHICS, DGL or equiv	97074/84A	97052A	92841A	92841A
GRAPHICS, AGP or equiv	97075/85A	97052A	92842A	92842A
IBM 3270	Yes	No	No	No
IBM 2780 RJE	Yes	No	91780A	No
IBM 3780 RJE	Yes	No	No	No
IMAGE	97073/83A	97053A	92069A	92069A
LAN (Local Area Net)	Yes	Yes	Yes	Yes
QUERY	Yes	Yes	92069A	92069A
MULTIPOINT	No	No	91730A	91750A
Sorting	sort	No	Yes	Yes
HP SPICE	98163/83A	No	92091A	No

I/O:

- . Direct access I/O using open, readir, writedir, seek, close
- . Sequential I/O operations append, close
- . Text I/O operations prompt, overprint, append, close
- . Can read into variables of type string & packed array of char
- . Can read into components of packed structures
- . Can read and write elements of enumerated types
- . Second parameter to reset, rewrite, open, and append to specify the physical file name
- . Third parameter to reset, rewrite, open, and append to specify the open options, i.e., CCTL, NCCTL

Statements, Program Control:

- . Constant expressions allowed in CASE label lists
- . Ranges allowed in CASE label list
- . OTHERWISE clause in CASE statements
- . Halt procedure

Expressions:

- . Literals may be of type LONGREAL ('E' is replaced by 'L' in the exponent)
- . Set denotations may be preceded by the name of the desired resulting set type

Heap Management:

- . Mark and release procedures added for disposing blocks of contiguous dynamic variables

2. Implementation Differences from HP Standard Pascal

HP1000 M,E,F (RTE-IVB, RTE-6/VM); A600, A700 (RTE- (RTE-XL)

- . Modules are not yet implemented
- . Strings are not yet implemented
- . Enumeration I/O is not yet implemented
- . Hex, octal, and binary are not yet implemented
- . Procedural parameters, when called, have access to the variables associated with the most recent activation of the procedures, not the variables which were associated with the procedure at the time that the procedure was passed.
- . PAC variables may have a lower bound <> 1
- . PAC expressions as well as literals are blank-filled before assignment, comparison, and value parameter passage if their lengths are less than that expected
- . The PACKED attribute does not distribute across dimensions in a multi-dimensional array, e.g.
 PACKED ARRAY [1..5, 1..5] OF T is equivalent to
 PACKED ARRAY [1..5] OF ARRAY [1..5] OF CHAR,
 not PACKED ARRAY [1..5] OF PACKED ARRAY [1..5] OF CHAR;
- . The following synonyms are not supported:
 (, for [and ,) for]
 (* for (@ and *) for)
 @ FOR ^

HP 9000 Under HP-UX

- . No modules at first release
- . No strings at first release
- . No hex, octal, & binary at first release
- . No enumeration I/O at first release
- . Blank-filling of PACs will be done as in Pascal/1000 (above)

HP 3000 Under MPE IV

- . Modules not yet implemented
- . Predefined routines (enabled by \$STANDARD_LEVEL 'HP3000'\$ option) assert, baddress, ccode, fnum, sizeof, waddress
- . Can specify a list of identifiers following a procedure heading e.g., PROCEDURE start; SPL EXTERNAL VARIABLE

HP 9826/9836

- . Strpos parameter list order is different
- . The \$UCSD\$ option causes the compiler to recognize most of the UCSD Pascal language
- . No append routine for files
- . No LINEPOS, OVERPRINT
- . No third parameter to reset, rewrite, and open to specify the open options

HP 3000

ALIAS
 ANSI
 ASSERT_HALT
 CHECK_ACTUAL_PARM
 CHECK_FORMAL_PARM
 CODE
 CODE_OFFSETS
 COPYRIGHT
 EXTERNAL
 GLOBAL

HEAP_COMPACT
 HEAP_DISPOSE
 INCLUDE
 LINES
 LIST
 LIST_CODE
 PAGE
 PARTIAL_EVAL
 PRIVATE_PROC
 RANGE

SEGMENT
 SKIP_TEXT
 SPLINTR
 STANDARD_LEVEL
 SUBPROGRAM
 TABLES
 TITLE
 USLINIT
 WIDTH
 XREF

HP 9826/9836

ALIAS
 ANSI
 CALLABS
 CODE
 CODE_OFFSETS
 COPYRIGHT
 DEBUG
 DEF
 HEAP_DISPOSE

IF
 INCLUDE
 IOCHECK
 LINENUM
 LINES
 LIST
 OVFLCHECK
 PAGE
 PAGEWIDTH

PARTIAL_EVAL
 RANGE
 REF
 SAVE_CONST
 SEARCH
 STACKCHECK
 SYSPROG
 TABLES
 UCSD

COMPETITIVE PASCAL FEATURE COMPARISONS

This page contains selected features available with competitive Pascal compilers and the HP9000.

Pascal Compilers						
		DEC	DATA GENERAL	PRIME	PERKIN- ELMER	HP 9000
	reentrant	yes			yes	yes
F	separate compilation	yes	yes	yes	yes	yes
E	direct access I/O	yes	yes			yes
A	produces re-entrant code			yes	yes	yes
T	routines in other langs.	yes	yes	yes	yes	yes
	structured constants					yes
R	flag non-standard source	yes				yes
E	global optimization	yes	yes	yes	yes	no
S	symbolic debugger	yes	yes	yes		no
	speed (lpm)	1500	>2500	?	400/1200	?
	price	5,500	6,000	5,000	5,250	6,000

* Blanks in the above chart indicate features that the vendors do not mention in their literature and thus we could not make any definite conclusions.



IMAGE/9000

IMAGE/9000

- * Same on BASIC and HP-UX
- * Query under BASIC only
- * "3000-like" Image

Enhancements:

- * Set, Record sizes unrestricted
- * Array data type allowed
- * Multi user (HP-UX)
- * Multi language access
- * Flexible buffering for excellent performance

CREATING THE DATA BASE	IMAGE/DAWN	IMAGE/	IMAGE/45	IMAGE/1000	IMAGE/3000
Data Base Capacity	20 volumes	836 Mbytes	266 Mbytes	960 Mbytes (200 on the L-XL)	960 Mbytes
Data Sets/Data Base	63 sets/base	50	32		99
Data Entries/Data Set	Disk limited	65 534	32 767	(32767**on the L-XL)	8 388 607
Data Entries/Chain	Disk limited	65 534	32 767	(32767**on the L-XL)	65 534
Data Entry Size (bytes)	Disk limited	1 022	1 022	4 096	4 096
Data Items/Data Entry	1023	127	127	127	127
Data Item Names/Base	1023	255	255	255	255
Can data item names be reused in different data sets?	Yes	Yes	Yes	Yes	Yes
Sorted Chains?	Yes	No	No	Yes	Yes
Compound Data Items?	Yes, multi-dimension arrays	Yes	Yes	Yes	Yes
Detail Data Sets/Master Search Items/Detail	31	8	16	16	16
Data Base Name (char.)	1-16	1-6	1-4	1-6	1-6
Data Set Name (char.)	1-16	1-15	1-15	1-6	1-16
Data Item Types	2 byte integer, 4 byte integer, 4 byte real, 8 byte real, 2 byte "code" ASCII multi-dimensional arrays of all types of above	1-word integer, 2-word real (5-digit BCD), 4-word real (12-digit BCD), ASCII (1022 max.)	1-word integer, 2-word real (6-digit BCD), 4-word real (12-digit BCD), ASCII (1022 max.)	1-word integer, 2-word real, ASCII (255 max.), 2-word integer, 4-word real	1-word integer, 2-word integer, 2-word real, 4-word real, 1-word logical ASCII (255 max.), Zoned decimal, Packed decimal
Max # of passwords/levelwords	13	31	31	15 levelwords	63
Level of security					
Data Set?	Yes	Yes	Yes	No	Yes
Data item?	No	No	No	Yes	Yes

*As many as can fit in any given cartridge

**Or a total of 4 Mbytes (including media records and all pointers) whichever is smaller

QUERYING THE DATA BASE	IMAGE/DAWN	IMAGE/250	IMAGE/45	IMAGE/1000	IMAGE/3000
QUERY Program Available?	Yes, on BASIC No, on UNIX	Yes	Yes	Yes	Yes
Batch capability?	Yes	No	No	Yes	Yes
Remote data base access?	No	No	No	Yes, between 1000 nodes	Yes, between 3000 nodes
Define new data base?	Yes	No	Yes	No	Yes
Add from form?	Yes	Yes	Yes	No	Yes
Add, Delete?	Yes	Yes	Yes	Yes	Yes
Replace?	Yes	Yes	Yes	Yes	Yes
Formatted reports?	1*	Yes	*1	#3	#3
Procedure files?	No	Yes	No	Yes	Yes
Helps?	Yes 2*	Yes	Yes	Yes	Yes
Data Base Information?	Yes	Yes	*2	Yes	Yes
Functional Operators	Yes	Yes	Yes	Yes	Yes
Whole Values	Yes	Yes	Yes	Yes	Yes
Partial Values	Yes	No	Yes	No	Yes
Computed Searches?	Yes	No	Yes	No	No
Sort for Report?	Yes	Yes	Yes	Yes	Yes
Threading (multiple set searches)	Yes	Yes	Yes	No	Yes
Browse?	Yes	No	Yes	No	Yes
Multiple level search?	Yes	No	Yes	No	Yes
QUERY front end to user program?	Yes	No	Yes	Yes	No

1* Just selective columnar and linear with no breaks of totaling

2* Can also print out diagram or schema

3* List by data item names in tabular format only

USER PROGRAM/HOST LANGUAGE ACCESS	IMAGE/DAWN	IMAGE/250	IMAGE/45	IMAGE/1000	IMAGE/3000
Multiple program access? Supported program languages	Yes BASIC PASCAL FORTRAN C UNIX	No BASIC	No BASIC	Yes PASCAL FORTRAN ASSEMBLER BASIC** Yes, between (from L- 1000 nodes node to 1000 node only)	Yes BASIC FORTRAN COBOL RPG II SPL Yes, between 3000 nodes
Remote Data Base Access DBOPEN	No	No	No		
Mode 1: modify, shared w/DB locking	Yes	Yes	No	Yes	Yes
2: Update, allow con- current update	No	No	No	No	Yes
3: Modify exclusively	Yes	Yes	Yes*	Yes*	Yes
4: Modify, allow con- current read	No	No	No	No	Yes
5: Read, allow con- current modify with locking	No	No	No	No	Yes
6: Ready, allow con- current modify	No	No	No	No	Yes
7: Read exclusive	No	No	No	No	Yes
8: Read shared	Yes	Yes	Yes	Yes	Yes
9: Modify shared with lock group locking	No	No	No	No	No
11: Modify exclusively, no mode to close	Yes	No	Yes	No	No
DBCLOSE					
Mode 1: Access to DB terminated	Yes	Yes	Yes	Yes	Yes
2: Referenced data set closed	Yes	No	No	Yes	Yes
3: Referenced data set rewound	Yes	Yes	No	No	Yes
4: Update Root File	Yes	Yes	Yes	No	No

*Buffers dumped

**Supported on 92069A IMAGE and QUERY Product Only

USER PROGRAM/HOST LANGUAGES ACCESS	IMAGE/DAWN	IMAGE/250	IMAGE/45	IMAGE/1000	IMAGE/3000
DBGET					
Mode 1: Reread	Yes	No	No	Yes	Yes
2: Serial read	Yes	Yes	Yes	Yes	Yes
3: Backward serial read	Yes	No	No	Yes	Yes
4: Directed read	Yes	Yes	Yes	Yes	Yes
5: Chained read	Yes	Yes	Yes	Yes	Yes
6: Backward chained read	Yes	No	No	Yes	Yes
7: Calculated read	Yes	Yes	Yes	Yes	Yes
8: Primary calculated read	No	No	No	No	Yes
DBINFO					
Mode 101: Item position security	Yes	Yes	Yes	Yes	Yes
102: Item descriptor	Yes	Yes	Yes	Yes	Yes
103: Items in a data base	Yes	No	No	Yes	Yes
104: Items in a data set	Yes	Yes	Yes	Yes	Yes
105: Items in a data set	Yes	-	-	-	-
106: Item null values	Yes	-	-	-	-
107: Pseudonym count, values	Yes	-	-	-	-
108: Code count, values	Yes	-	-	-	-
109: Picture word, null & range values	Yes	-	-	-	-
110: Dimensions, counts,sizes	Yes	-	-	-	-
201: Set position, security	Yes	Yes	Yes	Yes	Yes
202: Set description	Yes	Yes	Yes	Yes	Yes
203: Sets in a data base	Yes	Yes	Yes	Yes	Yes
204: Data sets using item	Yes	Yes	Yes	Yes	Yes
205: Sets list, set open status	Yes	-	-	-	-
206: Media record,buffer info	Yes	-	-	-	-
301: Paths in a data set	Yes	Yes	Yes	Yes	Yes
302: Primary search item	Yes	Yes	Yes	Yes	Yes
401: Data set Vol # status	No	Yes	Yes	Yes*	No
402: Volume name	No	Yes	Yes	Yes*	No
403: Volumes in DB status	No	Yes	Yes	No	No
404: Data sets on given volume	No	Yes	Yes	No	No
501: Item length & position in set	Yes	No	Yes	No	No
601: Volume number	Yes	-	-	-	-
602: Directory identifier	Yes	-	-	-	-
603: Directory path count, volume number	Yes	-	-	-	-
604: Set count, set numbers	Yes	-	-	-	-
605: ASCII path specifier	Yes	-	-	-	-

*Not the same as IMAGE/250 or IMAGE/45

USER PROGRAM/HOST LANGUAGE ACCESS	IMAGE/DAWN	IMAGE/250	IMAGE/45	IMAGE/1000	IMAGE/3000
DBLOCK					
Mode 1: Lock data base	Yes	Yes	No	Yes	Yes
2: Lock DB no wait	Yes	Yes*	No	Yes	Yes
3: Lock data set	Yes	Yes	No	Yes	Yes
4: Lock data set no wait	Yes	Yes	No	Yes	Yes
5: Predicate lock	No	Yes	No	Yes	Yes
6: Predicate lock no wait	No	Yes	No	Yes	Yes
7: Lock multiple sets	Yes	-	-	-	-
8: Lock multiple sets no wait	Yes	-	-	-	-
DBUNLOCK	Yes	Yes	No	Yes	Yes
DBUPDATE	Yes	Yes	Yes	Yes	Yes
DBPUT	Yes	Yes	Yes	Yes	Yes
DBDELETE	Yes	Yes	Yes	Yes	Yes
DBFIND	Yes	Yes	Yes	Yes	Yes
PACK USING/UNPACK USING/ PACK FMT	Yes	Yes	Yes	No	No

*Group Lock depends on Open Mode 9

MAINTAINING THE DATA BASE	IMAGE/DAWN	IMAGE/250	IMAGE/45	IMAGE/1000	IMAGE/3000
Transaction Logging & Recovery	No	-	No	Yes	Yes
DBSTORE/DBRESTORE	No	Yes	No	Yes	Yes
DBBACK-UP/DBRECOVER/TBKUP	Yes	No	Yes	No	No
DBLOAD/DBUNLOAD	Yes	Yes	Yes	Yes	Yes
DBERASE	Yes	Total DB or set-by-set basis	Yes	No	Yes
DBPURGE	Yes		Yes	No	Yes
DBCREATE	Yes		Yes	Not needed	Yes



HP 9000 - HP 1000 COMPARISON GUIDE

HP 9000 STRENGTHS	HP 1000 STRENGTHS
1. 32-Bit Address Space for Large Programs and Large Data Structures 2. Computational Performance Starts Where the 1000 Leaves Off. 3. Excellent Software Development Environment - hierarchical file system - UNIX shell commands - SCCS 4. Re-entrant Code Allows Shared Programs 5. IEEE Floating Point Format 6. Multi-CPU upgrade path	1. Real-Time Performance is Unsurpassed. 2. Hardware OEM's are supported with Documentation. 3. I/O Performance Can be Tuned - measurement & control I/O cards 4. State of the art DS capabilities 5. Power-Fail Battery Back-Up 6. 4Mb memory in A-Series

HP 9000 - HP 1000 Comparison Guide

Date: 16 Sep 1982

This document compares features and support for the HP 9000 HP-UX, HP 9000 BASIC Language System, HP 1000 RTE-6/VM, and HP 1000 RTE-A.1 operating systems.

Contents

Title	Description
Legend	Abbreviations used in this document
Computers	Models compared, CPU features
Memory	Systems, technology, performance
I/O System	Architecture, performance, features
Interfaces	Card types
Interface rates	I/O speeds for comparable interface cards
Disc drives	Models supported
Printers	Models supported, as TEXT printers
Graphics	Models supported, as GRAPHIC devices
Terminals	Models supported, as TEXT terminals
Mag tapes	Models supported, not just interfaceable
Miscellaneous	Cards, paper tape, measurement & control devices
Adaptors	Extenders, repeaters, muxes, modems, converters
Login/session	General features of user interface
User commands	List of comparable commands
Languages	Available & planned
Text editor	Types available, features compared
Operating system	System services and utilities
Program requests	System and utility program calls.
File system	Formats, capabilities compared
Feature products	Software packages available.

Legend

Term	Meaning
BLS	HP 9000 Basic Language System
CORONA	Corona configuration only.
CTU	264x Cartridge Tape Unit
DAWN	Dawn configuration only
DOUBLE	Double integer (2147483647)
DSN	Distributed Systems Network
FCS	Requires Factory Compiled Subprogram
HP+	Item available from OEM, Software Supplier, or user-group
MTS	Multipoint Terminal System
n.a.	Not Applicable to this system
No	The item is unsupported and known to NOT WORK.
Unsup	The item may work, but has not been tested.
User	Requires user-written code.
Yes	One or more versions of the item are supported.
###	Missing information
12345A	Number of product which supports this feature.

Any other entry identifies the feature or product supported.

Computers

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Models compared	Corona	Dawn	F-series	A-700
Model numbers	9030/40	9020	2117F	2137A
Architecture	Stack	Stack	Accumulator	Accumulator
Clock, cpu (MHz)	18	18	###	###
Clock, Real-Time				
Resolution (ms)	.5-500	.5-500	10	10
Operate w/power off	Yes	Yes	No	No
Control store (words*bits)	9k*38	9k*38	16k*32	16k*24
CPU's per system	1..3	1..3	1	1
Floating point imple.	Mixed	Mixed	Either	Either
Floating Point format	IEEE	IEEE	DSD	DSD
Instruction set size	230	230	~300	~250
Instructions, FFT	No	No	92835A	No
Instructions, scientific	Yes	Yes	2111/17F	A700
Instructions, vector	No	No	12824A	A700
Self-test code (bytes)	2300	2300	###	###
Technology	NMOS-III	NMOS-III	MSI-TTL	bit-slice
Time, CALL w/4 params (uSec)	3.3/7.7	3.3/7.7	7.75	8.75
Time, DOUBLE add (uSec)	.39	.39	4.6	3.0
Time, 64-bit F.P. add (uSec)	6.0	6.0	10.5	9.0
Time, micro-cycle (nSec)	55	55	175/280	250

Memory

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Battery backup	No	No	Yes	Yes
Bus rate (Mbytes/sec)	36	36	###	4.3
Chip density (k bytes)	128	128	64	64
Cycle time (fault control)	110ns	110ns	490ns	500ns
Cyle time (parity only)	n.a.	n.a.	420ns	454ns
2-bit Error in user prog	Halt	Halt	Down ptn	Down ptn
2-bit Error in system	Halt	Halt	Halt	Halt
Healing of memory at p-on	Yes	Yes	No	No
Maximum addressable (Mbytes)	537	537	2	4
Max. currently installable	2.5	2.5	2	4
Memory extender avail	No	No	Yes	No
Non-volatile memory (bytes)	2048+50	50	0	0
Power-fail trap	No	No	Yes	Yes
Program address space	Disc limit	All avail	64kb	64kb
Program size	Disc limit	All avail	404Mb	64kb
Technology	NMOS-III	NMOS-III	NMOS	NMOS
UCS/PCS (User ROM)	No	No	Yes	Yes
WCS (Writeble Control Store)	No	No	Yes	Yes

I/O System

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Device limit	Physical	Physical	255	63
Device addressing directly	No	Yes	No	No
Device addressing by LU	No	No	Yes	Yes
Device addressing by file	Yes	No	Yes	Yes
Device addressing by name	No	Yes	No	No
DMA channels	All	All	2	All
Fast HandShake I/O	No	No	Yes	n.a.
I/O, buffered	Yes	Yes	Optional	Optional
I/O, named buffers	###	Optional	No	No
I/O, unbuffered	Yes	No	Yes	Yes
I/O, circular buffers	Yes	Optional	No	No
I/O from microcode	No	No	Yes	n.a.
I/O status	Unsup	STATUS	EXEC(3/13)	EXEC(3/13)
I/O without wait	fork	TRANSFER	CLASS-I/O	CLASS-I/O
I/O without abort	###	Yes	Yes	Yes
I/O to bit bucket	Yes	Yes	Yes	Yes
Interrupt I/O	Yes	Yes	Yes	Yes
Interrupt schedule user code	Phase2	Yes	Yes	Yes
Interrupt latency (uSec)	30	30	75	30
Interrupt to driver (uSec)	125	125	200	140
Interrupt to user (mSec)	###	###	1..2	1..2
Interrupt simulation	Yes	Yes	No	No
IOP (I/O processor)	per 8 ch.	per 8 ch.	n.a.	per I/F
IOP code (bytes)	###	###	0	###
Locking of interfaces	No	Internal	Internal	Internal
Locking of devices	Phase2	No	Yes	Yes
Lock passing to other prog	###	No	Yes	Yes
Maximum I/O slots	23	20	46	16
Rate, I/O bus (Mb/sec)	n.a.	n.a.	2.2	4.3
Rate, IOP (Mb/sec)	5.1	5.1	n.a.	2.0
Rate, I/O slot, burst (Mb)	5.1	5.1	11 (uCode)	###
Rate, I/O slot, DMA/Muxed	0.5/1.0	0.5/1.0	1.0	2.0
Rate, I/O slot, intr/sec	###	###	4000	###
Re-directed I/O	Yes	Yes	Yes	Yes
Register access	Unsup	Some	Some	Some
Single char/word I/O	###	Yes	Yes	Yes
Spooling output	Printers	No	Yes	Yes
Spooling input	No	No	Yes	No
User written interface dvr	No	No	Yes	Yes
User written device driver	No	No	Yes	Yes
User-designed interfaces	No	No	Yes	Yes

I/O Interfaces

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Analog (D/A out, A/D in)				
Input	No	No	91000A	12060A
Multiplexed input	No	No	No	12061A
Output	No	No	12555B(obs)	12062A

ASI, BACI, TTY (Asynchronous Serial)

Current loop	27128A	27128A	12531C/D	12005B
Enq/Ack (h=host only)	27128A	27128A	12966A(h)	12005B(h)
Multiplexed RS-232 (ch)	27130A(8)	Unsup	12792A(8)	12040A(8)
Multiplexed RS-449	#####	#####	12792A	12040A
Multiplexed modem	No	No	12620A(16)	No
RS-232C	27128A	27128A	12531x/966A	12005B
RS-232C Modem	27128A	27128A	12531x/966A	12005B
RS-423/449	27128A	27128A	No	12005B
Xon/Xoff	27128A	27128A	No	No

DSN (Distributed Systems Network)

BISYNC-DIRECT	No	No	12834A	12082A
BISYNC-MODEM	No	No	12793A	12073A
DL (MTS BISYNC host)	No	No	12790A	No
DL (MTS BISYNC slave)	No	No	12830A	12072A
HDLC-DIRECT	No	No	12825A	12044A
HDLC-MODEM	No	No	12794B	12007B
HSI (DS-3000)	No	No	12889A	No
RJE (IBM BISYNC)	27122A	Unsup	12618A	No
SRM (Ganglia)	#####	#####	No	No
WASP Hardwired (DS/1000)	No	No	12771A	No
WASP Modem (DS/1000)	No	No	12773A	No
X.25 (LAP-B)	No	No	12250A	12075A

GPIO, PIC (Parallel)

BCD (Binary Coded Decimal)	No	No	12556B	No
8-bit	27112A	27112A	12597A	12006A
16-bit single ended	27112A	27112A	12554A/66B	12006A
16-bit differential	No	No	12930A	12063A
16-bit for 2607/13/17/19	No	No	12845B	No
16-bit for 2608A	No	No	26099A	No
16-bit for 6940B	No	No	14550A	No
16-bit for 7970B/E	No	No	13181/83B	No
16-bit isolated	No	No	No	12063A
16-bit relay	No	No	12551B	No

HP-IB (IEEE-488/1978)

Instrument (488-1975)	27110A	27110A	59310B	12009A
Disc (488-1978)	27110A	27110A	12821A	12009A
Extender on card	No	No	No	37203L

Miscellaneous

Automatic Calling Unit	No	No	12589A	No
Clock	Internal	Internal	12539C	Internal
CRT Monitor	97062A	97062A	91200B	No

OEM

Breadboard	No	No	12620A	12010A
PROM	No	No	No	12008A
PSI (Programmable Serial)	#####	#####	12826B	12042B

Service

Logic Analyzer	No	No	10284B	10285A
Service extender	27116A	27116A	02116-...	12001A
Slot jumper	n.a.	n.a.	12777A	12012A

Interface Rates

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Async Mux, per channel	###	###	19.2K	19.2K
Async Mux, aggregate (kb/s)	###	###	76.8	76.8
Async Terminal (K bytes/sec)	19.2	19.2	9.6	56
HP-IB	1M	1M	1M	1M
Parallel, 16-bit (Mb/sec)	.95	.95	1.0	1.1
Parallel, 8-bit	.48	.48	1.0	###
PSI sync, Kb/sec	###	###	500	500
PSI async, Kb/sec	57.6	###	57.6	57.6

Disc Drives

	[command set]	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
12732A	9885 for 100C	No	as 9885	Yes	No
13272A	9130A for 2642/47F	No	No	as CTU	as CTU
7900A	2Mb ctg	No	No	Yes	No
7902A/K	8" flex [9895]	Unsup	Unsup	Unsup	Unsup
7905A	15 Mb ctg [MAC]	No	No	Yes	No
7905A	HP-IB 7905 [AMIGO]	No	Unsup	No	No
7906H	20 Mb ctg [ICD]	No	No	Yes	Yes
7906M/S	20 Mb ctg [MAC]	No	No	Yes	No
7906M/S	HP-IB 7906 [AMIGO]	No	Unsup	No	No
7908P	16 Mb winch [CS80]	Yes	Yes	Yes	Yes
7908R	rack 7980P [CS80]	Unsup	Unsup	Yes	Yes
7910H	12 Mb winch [ICD]	No	Unsup	Unsup	Yes
7911P	27 Mb winch [CS80]	Yes	Yes	Yes	Yes
7911R	rack 7911P [CS80]	Unsup	Unsup	Yes	Yes
7912P	64 Mb winch [CS80]	Yes	Yes	Yes	Yes
7912R	rack 7912P [CS80]	Unsup	Unsup	Yes	Yes
7920H	50 Mb pack [ICD]	No	No	Yes	Yes
7920M/S	50 Mb pack [MAC]	No	No	Yes	No
7920M/S	HP-IB 7920 [AMIGO]	No	Unsup	No	No
7925H	120 Mb pack [ICD]	No	No	Yes	Yes
7925M/S	120 Mb pack [MAC]	No	No	Yes	No
7925M/S	HP-IB 7925 [AMIGO]	No	Unsup	No	No
7933H	404 Mb fixed [CS80]	Yes	Yes	Unsup	Unsup
82901M	5" flex dual [9895]	Yes	Yes	No	No
82902M	5" flex sngl [9895]	Yes	Yes	No	No
9130A	5" flex sngl	No	No	No	No
9130K	internal 9130A	Yes	Yes	n.a.	Model 6 only
9134A	5Mb winch [9895]	Unsup	Unsup	Yes	Yes
9135A	9134A+9130K [9895]	Unsup	Unsup	No	No
9138A	9134A+9895A [9895]	Unsup	Unsup	Yes	Yes
97093A	10Mb Seagate 412	Yes	Yes	n.a.	n.a.
9867/80A	7900A for 9800	No	No	No	No
9885M/S	8" flexible	No	Yes	Yes	No
9895A	8" flexible [9895]	Yes	Yes	Yes	Yes

Printers

		HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
2601A	40 cps daisywheel	Unsup	Unsup	Yes	Yes
2607A	200 lpm impact	No	No	Yes	No
2608A	400 lpm impact				
2608A	HP-IB	Unsup	Unsup	Unsup	Yes
2608A	16-bit parallel	No	No	Yes	No
2608S	400 lpm impact				
2608S	HP-IB CIPER	Unsup	Unsup	Unsup	Yes
2608S	MTS CIPER	No	No	Yes	No
2611A	600 lpm chain	###	###	###	###
2613A	300 lpm drum	No	No	Yes	No
2617A	600 lpm drum	No	No	Yes	No
2619A	1000 lpm chain	No	No	Yes	No
2631B/G	180 cps impact				
2631B/G	HP-IB	Yes	Yes	Unsup	Yes
2631B	RS-232-C	###	###	Unsup	Yes
2631B	16-bit parallel	No	No	Yes	No
2631B	8-bit w/264x	No	No	Yes	No
2671A/G	120 cps thermal	Unsup	Unsup	Yes	Yes
2673A	120 cps thermal				
267x	HP-IB	Unsup	Unsup	Unsup	Yes
267x	RS-232-C	Unsup	Unsup	Unsup	Yes
267x	8-bit w/264x	No	No	Yes	No
2680A	45 ppm laser	Unsup	Unsup	Unsup	Unsup
7240A	40 cps thermal	Unsup	Unsup	Unsup	Unsup
7245A/B	40 cps thermal	Unsup	Unsup	Yes	Unsup
7310A	500 lpm thermal	Unsup	Unsup	Unsup	Unsup
82905A/B	80 cps impact	Unsup	Unsup	Unsup	Yes
97090A	480 lpm internal	Yes	Yes	n.a.	n.a.
9861A	30 cps typing	No	No	No	No
9866A	200 lpm thermal	Unsup	Unsup	Yes	No
9871A	30 cps daisywheel	Unsup	Unsup	Yes	Unsup
9876A	480 lpm thermal				
9876A	HP-IB	Unsup	Unsup	Unsup	YES
9876A	RS-232C	Unsup	Unsup	Unsup	Unsup
9876A	8-bit	No	No	No	No
9881A	2607A for 9800	No	No	No	No

Graphics peripherals

Note: This section discusses support for these peripherals as graphics devices. "Unsup" means that the host graphics software supports the device's command set but has not been tested with this specific device. "No" means that there is no support for the device's command set and there may be no interface connection. "Rdump" means that a BASIC "DUMP GRAPHICS" command may work. The number of screen pixels and printer pixels may mismatch.

		HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
1345A	16cm binary CRT	No	No	No	No
1347A	16cm HP-GL CRT	Unsup	Unsup	###	###
1350A/S	2k translator				
1351A/S	8k translator				
1350/51	HP-IB (not HP-GL)	###	No	Yes	Yes
1350/51	RS-232-C	###	No	###	###
1350/51	16-bit parallel	No	No	No	No

1360A	system	No	No	No	No
2608A	Boise raster				
2608A	16-bit parallel	No	No	Yes	No
2608A	HP-IB bin raster	No	No	No	###
2608S	HP raster				
2608S	HP-IB, HP raster	###	Rdump	No	Yes
2608S	MTS	###	No	Yes	No
2623A/29G	DTD Esc* CRT	Yes	No	Yes	Yes
2631G	HP raster	###	Rdump	###	###
2647A	DTD Esc* CRT	Unsup	No	Yes	Yes
2647F	Floppy 2647A	Yes	No	###	###
2648A/49G	DTD Esc* CRT	Unsup	No	Yes	Yes
2649C	same as 2647	As 2647	No	Yes	Yes
2671G	HP raster				
2673A	HP raster				
267x	HP-IB, HP raster	###	Rdump	Unsup	Unsup
267x	RS-232-C	###	Rdump	Unsup	UNsup
267x	8 & 16-bit	No	No	No	No
2680A	### laser	###	###	No	No
2703A	Orion color	Yes	No	###	###
7210A	1 pen parallel	No	No	Unsup	No
7220A/B/C	HP-GL RS-232-C	Unsup	Unsup	Unsup	Unsup
7220S/T	7220 paper feed	Unsup	Unsup	Unsup	Unsup
7221A/B/C	SDD Esc. RS-232C	###	No	Yes	Yes
7221S/T	7221 paper feed	###	No	Yes	Yes
7225A/B	1 pen HP-GL	Unsup	Unsup	Yes	Yes
7240A	HP-GL serial	Unsup	Unsup	Unsup	Unsup
7245A/B	HP-GL HP-IB	Unsup	Rdump	Yes	Yes
7310A	HP raster	Unsup	Rdump	###	###
7470A	HP-GL 2-pen	Yes	Yes	Yes	Yes
7580A	HP-GL D-size				
7585A	HP-GL E-size				
758x	HP-IB	Yes	Yes	Yes	Yes
758x	RS-232-C	Unsup	###	Yes	Yes
82905A	Epson raster	No	No	No	No
82905B	HP raster	###	Rdump	###	###
9111A/T	HP-GL tablet	Yes	Yes	Yes	Yes
91200A	RS-170 video	n.a.	n.a.	Yes	No
97062A	CRT monitor RGB	Yes	Yes	n.a.	n.a.
97090A	Int HP raster prt	###	Rdump	n.a.	n.a.
9862A	parallel 1-pen	No	No	No	No
9872A/B/C	HP-GL, HP-IB	Yes	Yes	Yes	Yes
9872S/T	9872 paper feed	Yes	Yes	Yes	Yes
9874A	Digitizer	Unsup	Unsup	Yes	Yes
9876A	HP raster	###	Rdump	###	###
98770B	AURORA	Yes	Yes	n.a.	n.a.
98775A	Light pen	Yes	Yes	n.a.	n.a.
98780B	ODYSSEY	Yes	Yes	n.a.	n.a.

Terminals

		HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
2382A	Office	Unsup	Unsup	Yes	Yes
2600A	Beehive	Unsup	Unsup	Yes	Unsup
2615A	Hazeltine	Unsup	Unsup	Yes	Unsup
262x	printer pass-thru	User code	User code	Yes	No
2621A/B	Z80 char/line mode	Unsup	Unsup	Yes	Yes
2622A	Block mode 2621	Yes	Unsup	Yes	Yes
2623A	Graphic 2622	Yes	Unsup	Yes	Yes
2624A/B	Dual-port 2622				
2624	RS-232-C	Unsup	Unsup	Yes	Yes
2624B	MTS	No	No	Yes	No
2626A	Windowed 2624				
2626A	RS-232-C	Unsup	Unsup	Yes	Yes
2626A	MTS	No	No	Yes	No
2626W	HP WORD 2626A	as 2626A	as 2626A	as 2626A	as 2626A
2629x	OEM 262x	as 262x	as 262x	as 262x	as 262x
2635A/B	printing	Unsup	Unsup	Yes	Yes
2639A	OEM 2635A	Unsup	Unsup	Unsup	Unsup
264x	RS-232-C	Unsup	Unsup	Yes	Yes
264x	MTS	No	No	Yes	No
264x/13238A	prtr pass-thru	User-code	User-code	Yes	No
264x/13250B	prtr pass-thru	User-code	User-code	Yes	No
264x/13296A	HP-IB pass-thru	User-code	User-code	User-code	User-code
2640A/B	8008 line/blk mode	Unsup	Unsup	Yes	Yes
2641A	APL 2645	as 2645	as 2645	as 2645	as 2645
2642A	W/P 2645 w/floppy	Unsup	Unsup	Yes	Yes
2644A	2640 w/CTU	Unsup	Unsup	Yes	Unsup
2645A	8080 2644A	Unsup	Unsup	Yes	Yes
2647A	BASIC 2648	Unsup	Unsup	Yes	Yes
2647F	8085 2642A+2647A	Yes	Unsup	###	###
2648A	2645 w/graphics	Unsup	Unsup	Yes	Yes
2649x	OEM 264x	as 264x	as 264x	as 264x	as 264x
2675A	Thermal printing	Unsup	Unsup	Yes	Unsup
270xA	Orion	Yes	Unsup	###	###
275x/6x	TTY's, Terminettes	No	No	Yes	No
307x	RS-232-C	Unsup	Unsup	Yes	Yes
307x	Serial Data Link	No	No	Yes	No
307x	MTS (FDL)	No	No	Yes	No
3070A/B	Datacap Serial DL	No	No	Yes	No
3071A	Datacap RS-232C	Unsup	Unsup	Yes	RS-232
3075..7	Datacap	Unsup	Unsup	Yes	RS-232

Magnetic Tape

		HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
7970B	800 bpi NRZI	No	No	Yes	No
7970E	1600 bpi PE				
7970E	HP-IB	Yes	Yes	No	Yes
7970E	16-bit parallel	No	No	Yes	No
7971A	7970+cabinet	as 7970E	as 7970E	Yes	as 7970E
7976A	6250 bpi GCR	No	No	Special	No
88140L/S	DC-600A in 79xx	Yes	Yes	Yes	Yes
98200A	DC-100A cartridge	No	No	264x only	264x only
9865A	Cassette	No	No	No	No
9875A	DC-100A SIF format	No	User-code	User-code	User-code
9877A	DC-100A for 9800	No	No	No	No

Miscellaneous peripherals

		HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
1000AP	1000L in 2250A	No	No	Yes	Yes
2240A	RVD M&C processor	Unsup	Unsup	Yes	Yes
2250A	RVD M&C processor	Unsup	Unsup	Yes	Yes
2748A/B	Paper tape reader	Unsup	Unsup	Yes	Yes
2894A	600 cpm card reader	No	No	Yes	No
2895A/B	Paper tape punch	Unsup	Unsup	Yes	Yes
3078A	MTS data coupler	No	No	Yes	No
7260A	RS-232 card reader	Unsup	Unsup	Unsup	Unsup
7261A	16-bit 7260	Unsup	Unsup	Yes	Unsup
9860A	Single card reader	No	No	No	No
9869A	8-bit 7260	No	No	No	No
9870A	Advanced 9860	No	No	No	No
6940B	NJD CAT box	Unsup	Unsup	Yes	Yes
6942A	NJD CAT processor	Unsup	Unsup	Yes	Yes

I/O Adaptors & Extenders

		HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
I/O Extender		#####	97098A	12979A/B	No
13264A	26xx-Data Link(DL)	No	No	Yes	Yes
13265A	26xx-Modem	Yes	Yes	Yes	Yes
13266A	26xx-Current Loop	No	No	Yes	Yes
13267A	26xx-Async MTS	No	No	Yes	No
13268A	26xx-Sync MTS	No	No	Yes	No
26075A	HP-IB switch	Unsup	Unsup	Unsup	Unsup
30037A	Async repeater	Unsup	Unsup	Yes	Unsup
3074A/M	RS-232C to DL	No	No	Yes	Yes
37201A	RS-232 HP-IB ext.	###	###	Yes	Yes
37203A	Coax/fibre HP-IB	###	###	Yes	Yes
37203L	1000A/L 37203A	No	No	No	Yes
37230A	Short haul modem	###	###	Yes	Yes
39301A	Fibre optic mux	Yes	Unsup	Unsup	Unsup
59403A	HP-IB CCI	Unsup	Unsup	Unsup	Unsup
9868A	98x0 I/O expander	No	No	No	No
9878A	98x5 I/O extender	No	No	No	No
9888A	98x6 Bus Expander	No	No	No	No

Login/Session Features

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Accounting of users/groups	Phase2	No	Yes	No
Broadcast to all users	Yes	No	Yes	No
Command capability checking	No	No	Yes	No
Errors, sentence messages	No	Optional	###	Optional
HELP file/facility	Yes	No	Yes	No
Launch login @ another term	\$ct(?)	n.a.	No	n.a.
Mail (user-user message)	Yes	No	Yes	No
Manual, on-line	Yes	No	No	No
Password at login	Yes	No	Yes	No
Super-user or system manager	Yes	n.a.	Yes	n.a.
Time, user connected	Yes	n.a.	Yes	No
Time, cpu used by user	Yes	n.a.	Yes	No
Users, absolute maximum	###	1	99	62

User Commands/Utilities

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Command levels:				
Kernel	###	BASIC	RTE(*)	RTE(*)
Intrinsic	###	BASIC	RTE(*)	CMND(:)
Interpreter	sh(\$)	BASIC	FMGR(:)	FMGR(:)
Alarm/Break/Signal program	###	EVENT	Break	Break
Backup by date	\$dump	No	No	No
Calculator emulate	\$bc,\$dc	Yes	:CA/:DP	:CA/:DP
Calendar memo	\$calendar	No	No	No
Case folding	Yes	No	Yes	Yes
Command entry stack	###	###	Yes	No
Command stream branch	\$test	No	:IF	:IF
Command stream files	Yes	No	:TR	:TR
Command stream nesting	###	n.a.	13	13
Command timed execute	\$cron	BASIC	No	No
Directory change	\$cd	MSI	No	n.a.
Directory compare	\$dircmp	No	No	No
Directory create	\$mkdir	CREATE DIR	No	No
Directory extend?	Yes	SDF only	No	No
Directory list	\$ls	CAT	:DL	:DL
Directory list partial name	\$ls/??	Yes	:DL,nam--	:DL,nam--
Directory list wild cards	\$ls/??	No	:DL,x-x-	:DL,x-x-
Directory purge	\$rmdir	PURGE	:IN	:IN
File compare	\$cmp,\$diff	No	HP+	HP+
File copy	\$cp	COPY	:ST	:ST
File create	Yes	CREATE...	:CR	:CR
File dump	\$pr,\$cat	No	:DU	:DU
File equations	\$dd	ASSIGN	No	No
File list, ASCII, numbered	\$catn	No	:LI,,A	:LI,,A
File list, octal	\$od	No	:LI,,B	:LI,,B
File purge	\$rm	PURGE...	:PU	:PU
File unpurge	No	No	User	User
Implied run	Yes	No	Yes	Yes
Messages to operator	write,mail	n.a.	:AN	:AN
Messages to other user	write,mail	n.a.	:SM	n.a.
Message for me, display	write,mail	n.a.	:ME	n.a.
Message for me, to file	###	n.a.	:ME	n.a.
Print banner	\$baner	demo	HP+	HP+
Print calendar	\$cal	No	No	No
Print primes	\$primes	demo	HP+	HP+

Program start	Yes	RUN	:RU/*ON	:RU/*ON
Re-direct I/O	<,>	@Name	:SL/:CS	:SL/:CS
Shutdown system	shutdown	STOP ALL	*SD,0	No
Sub-directories	Yes	Yes	No	No
Time set	date	SET TIME..	*TM	:TM
Time show	date	TIME\$	*TI	:TM
Timeslice examine/change	###	No	*QU	:QU
Type ahead buffer	Yes	Live kbd	MUX only	MUX only
User defined commands	Yes	No	No	No
Volume copy	###	No	:CO	:CO
Volume dismount	\$umount	VDA for DB	:DC	:DC
Volume initialize	###	INITIALIZE	:IN	:IN
Volume mount	\$mount	VDA for DB	:MC	:MC
Volume pack	###	Utility	:PK	:PK
Volume space	\$df	CAT	:DL	:DL

Computer
Museum

Languages

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Assembly programming	No	No	ASMB	No
HP BASIC, compiled	No	92050A	No	No
HP BASIC, interpreted	No	92050A	92101A	92076A
bc (binary calculator)	###	n.a.	No	No
bs (Bell BASIC)	###	No	No	No
HP C	Yes	No	HP+	###
COBOL	No	No	HP+	###
Command interpreter progs	bc,dc	No	limited	limited
efl (BELL extended ftn)	###	No	No	No
FORTTRAN IV, 66, 4X	No	No	92834A(4X)	No
FORTTRAN 77	97071/81A	No	92836A	92836A
Macro assembly	No	No	MACRO	MACRO
Microprogramming	No	No	92061A	92045A
HP Pascal	97072/82A	No	92832A	92833A
PL/1	###	No	No	No
RATFOR	###	No	HP+	###
RPG	###	No	No	No
SNOBOL	###	No	No	No

Text Editing

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Line editor	ed,ex	n.a.	EDITR	EDIT/1000
Screen editor	vi	EDIT	EDIT/1000	EDIT/1000
Batch edit command files	ed,bfs	No	Yes	Yes
Block text copy	Yes	Yes	Yes	Yes
Block text move	Yes	Yes	Yes	Yes
Case folding	Yes	No	Yes	Yes
Column windowing	No	No	Yes	Yes
Cut fields	cut	No	No	No
Display functions mode	###	Always	Optional	Optional
Edit non-language files	Yes	No	Yes	Yes
Find	Yes	Yes	Yes	Yes

Global search and replace	Yes	Yes	Yes	Yes
Help	H	No	Yes	Yes
Output formatter/typesetter	nroff/troff	No	HP+	###
Post during edit	Yes	Yes	Yes	Yes
Regular expressions	Yes	No	Yes	Yes
Screen copy (off-line edit)	###	n.a.	Yes	Yes
Spelling/typo package	Yes	No	###	###
Structured program indent	cb	Yes	No	No
System commands from edit	Yes	Yes	Yes	Yes
Time stamp within text				
Automatic in special field	No	No	Yes	Yes
On command	Yes	No	Yes	Yes
Work area in memory/on disc	DISC	MEMORY	DISC	DISC
Work files named	Yes	n.a.	Yes	Yes

Operating System Features

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Backup of disc to disc/tape:				
Single file	###	COPY	:ST/FCOPY	:ST/FCOPY
Selected files	###	###	FCOPY,SAVER	FCOPY,SAVER
Related files	###	###	FCOPY	FCOPY
Entire directory	###	###	FCOPY	FCOPY
Entire volume	###	###	FCOPY,WRITT	FCOPY,WRITT
Physical disc area	###	###	!DISK	###
Physical disc volume	###	###	!DISK	###
Code:				
Re-entrant	Yes	Yes	Some	Some
Recursive	Yes	Yes	Some	Some
Separate from data	Yes	Yes	No	No
Shared among users	Shared	No	Cloned	Cloned
in Virtual memory	Segmented	No	Segmented	No
Data:				
in local COMMON	Yes	Yes	Yes	Yes
in named local COMMON	Yes	Yes	Yes	Yes
in shared system memory	MEMORYfile	MEMORYfile	EMA	EMA
in system COMMON	###	No	Yes	Yes
in system named COMMON	###	No	SSGA	Yes
address limit (Mbytes)	Disc	Memory	Disc	12.6
in virtual memory	Paged	No	Paged	Yes
Debug:				
program abort core dump	Yes	No	No	No
program profiler	Yes	No	92083A	No
program absolute debug	###	No	DBUGR	DBUGR
program symbolic debug	adb	No	No	No
program source debug	###	Yes	HP+	HP+
system abort core dump	Yes	No	Unsup	VCP
system absolute debug	###	No	CDA6	LMOD
system symbolic debug	crash	No	###	###
system profiler	Yes	No	Unsup	Unsup
Entry point name lengths:				
Local (user program)	Lang dep.	15	Lang dep.	Lang dep.
Global (system wide)	14	16	5/15	5/15
Error log file	Yes	No	No	No
Library management:				
binary code	ar	COMPILE	MERGE/INDEX	MERGE/IND
source code	SCCS	No	No	No

Partitions:				
dynamic size	###	Yes	No	Yes
maximum number	###	60	64	255
maximum size	###	All avail	All avail	All avail
Priorities:				
partitions	n.a.	5+2(INTR)	RT/BG	RT/BG
programs	19	n.a.	32767	32767
tasks within program	n.a.	16	n.a.	n.a.
user/group	###	n.a.	n.a.	n.a.
Programs:				
core-resident	Yes	Always	Yes	Yes
max number active	###	60	255	255
swappable	Yes	No	Yes	Yes
segmentable	Automatic	LOAD SUB	Automatic	EXEC(8)
Swapping of idle programs	Yes	No	Yes	Yes
System generator	config	###	RT6GN	RTAGN
Time share/slice	Share	Share	Slice	Slice

Requests from a Program

	HP-UX	9000 BLS	RTE-6/VM	RTE-A.1
Dynamically allocate memory	###	ALLOCATE	No	No
Fetch user's run command	###	No	EXEC(14)	EXEC(14)
I/O requests:				
cancel pending read	###	Unsup	CLRQ	CLRQ
read	###	many	EXEC(1)	EXEC(1)
status control	###	CONTROL	EXEC(3)	EXEC(3)
status read	###	STATUS	EXEC(13)	EXEC(13)
write	###	many	EXEC(2)	EXEC(2)
Load segment/overlay	###	LOAD SUB	EXEC(8)	EXEC(8)
Lock self in memory	###	n.a.	EXEC(22)	EXEC(22)
Memory available to self	###	###	EXEC(26)	EXEC(26)
Partition status	###	PTN..STAT	EXEC(25)	EXEC(25)
Pass command from program	Yes	OUTPUT KBD	MESSS	MESSS
Peek and Poke memory	Yes	No	User	User
Process status	Yes	PTN..STAT	User	User
Program-to-program:				
I/O	Pipes	MEMORYfile	CLASS I/O	CLASS I/O
Signals	Yes	EVENT	RNRQ	RNRQ
Suspend	Yes	No	User	User
Abort	Yes	No	EXEC(6)	EXEC(6)
Request time/date	Yes	TIME/DATE	EXEC(11)	EXEC(11)
Schedule other program to run:				
At a specified Time	Yes	User	EXEC(12)	EXEC(12)
Now, but only if idle	###	CREATE PTN	EXEC(9,10)	EXEC(9,10)
Now or when next idle	###	No	EXEC(23,24)	EXEC(23,24)
and suspend caller	###	No	EXEC(9,23)	EXEC(9,23)
w/o suspending caller	###	CREATE PTN	EXEC(10,24)	EXEC(10,24)
Suspend self:				
Indefinitely (PAUSE)	Yes	PAUSE	EXEC(7)	EXEC(7)
For a specified time	Yes	WAIT	EXEC(12)	EXEC(12)
Until a specified time	###	ON TIME	EXEC(12)	EXEC(12)
Until a specified date	###	ON TIMEDATE	No	No
Until other program stops	Yes	User	User	User
Turn off system protection	No	No	\$LIBR	\$LIBR

1. HP Standard Pascal Differences from ISO/ANSI standards

HP Standard Pascal allows:

General:

- . ASCII character set
- . Compiler options of the form \$<option>, <option>, ... \$
- . Underscore character ('_') allowed in identifiers
- . Identifiers may be up to a source line in length
- . Predefined constants maxint and minit define the largest and smallest integer respectively
- . Predefined functions hex, octal, binary (convert a string, or PAC (Packed Array of Characters) containing digits in base 16, 8, and 2 respectively, to an integer)

Compilation units:

- . Allows a collection of routines with no main program body to be compiled separately from the main program
- . Allows modules to be defined to facilitate program structuring

Routines:

- . Allows FORWARD and EXTERNAL procedures and functions
- . Functions can return structured types

Declarations:

- . CONST, TYPE, and VAR sections may appear in any order, multiple occurrences per block
- . Constant expressions allowed in CONST section
- . Constant expressions allowed in subrange definitions
- . Allows structured constants (constants of type array, record, set, string and PAC). ISO/ANSI allows a limited form of set and PAC constants

Types:

- . Predefined type LONGREAL
- . Strings
 - Syntax for non-printing characters in string literals
 - String literals padded with blanks in assignments and comparisons
 - Can access a component character of a constant string
 - Can assign, compare, pass strings with different current and max lengths
 - Conversions between strings and Packed Arrays of Chars
 - Predefined procedures: strappened, strread, strwrite, setstrlen, strmove, strinsert, strdelete
 - Predefined functions: strlen, strmax, strpos, strltrim, strrtrim, strrpt, str

GLOSSARY OF TERMS

(This list does not include any of the HP-IO terms yet.)

BACKGROUND PARTITION- the partition(s) that does not have the keyboard or physical CRT resources attached to it.

CACHE MEMORY- a mechanism interposed in the memory hierarchy between main memory and the CPU to improve effective memory transfer rates.

COMPILED CODE- see object code

CONTEXT- A set of related addresses that define a scope of addressability, i.e., limit the set of code and/or data addresses which are accessible. Examples: programs, processes, partitions.

DIRECT ADDRESSING- the location of an operand or a branch target may be indicated by a convention in which a base register and offset are specified explicitly in the instruction.

DISPATCHER- The dispatcher is responsible for the short term scheduling of the processor resources.

EVENT- a semaphore that provides synchronization of partitions.

EXTERNAL DATA SEGMENT (EDS)- an arbitrary additional data segment which stores variables that are not local to program procedures or parameters to those procedures. External data segments are allocated as a result of explicit requests from the operating system (SUN).

FLOATING POINT STANDARD (IEEE)- The IEEE floating point standard proposal defines 4 floating point formats. These are:

- single 32 bit format
- single-extended format
- double 64 bit format
- double-extended format

The proposal states that all implementations must support the single format and should support the extended format corresponding to the widest basic format, 32 or 64-bit. The HP 9000 family supports the single 32 bit format and the double 64 bit format.

FOREGROUND PARTITION- the partition that possesses the physical CRT and keyboard resources. There can only be one foreground partition.

GLOBAL VARIABLES- variables which are accessible throughout the program. This is opposed to a local, which is accessible only within the environment it is created.

GLOBAL DATA SEGMENT- a data segment which stores program variables which are not local to procedures or parameters to those procedures.

INDIRECT ADDRESSING- this addressing mode is either indicated by a bit in the instruction or the instruction. In the first case, the base and the offset information in the instruction is utilized as it is in the direct case to fetch a word from memory. In the indirect case, this word is not the desired operand, but is a pointer to the desired operand. The four pointer formats used are self-relative pointers, stack pointers, global data segment pointers, and external data segment pointers.

INTERMEDIATE CODE- internal program code from which listings, tracing, etc., can be generated.

KERNEL- The software modules of the operating system that manage the allocation and deallocation of resources, i.e. memory, processors.

LOCAL VARIABLE- a variable which is accessible only within the subprogram environment it is created, as opposed to a global variable which can be accessed throughout the program.

MULTI-PROCESSING- the ability to execute multiple processes concurrently.

MULTI-TASKING- the ability to execute multiple tasks concurrently.

NON-VOLATILE MEMORY- memory that does not lose its contents when power is turned off.



OBJECT CODE- compiled code in the final machine instructions which the machine performs while executing.

PAGE- The unit of storage utilized by a system which directly dictates the form of addresses, as well as the organization of programs and/or data.

PARTITION CONTEXT- A contiguous set of mapped memory blocks. The addresses onto which memory has been mapped defines the scope of addressability of a partition context. Size can vary from 16Kb to 8Mb. Partition contexts provide the physical (real) memory for an arbitrary number of process.

PRIMITIVE- the most elementary operations that synchronize and control processes using semaphores.

PROCEDURE CONTEXT- a set of instructions which make up a program.

PROCESS CONTEXT- The unique instance of an executing program. The minimum process consists of the program context, stack and global data segments, and the current hardware state. The size and segment membership can be dynamically varied.

PROGRAM CONTEXT- A set of one or more procedures, each procedure being a collection of instructions with a common entry name, which may or may not be parameterized. A program may occupy a partial, one or many code segments. The scope of addressability is the set of code segments which it occupies. A program is a static context.

QUEUE- a data structure for the temporary storage of information that uses either the last in, first out algorithm or the first in, first out algorithm.

SEGMENT- a logical grouping of information, such as a program, subprogram, array, or data area. A segment may or may not consist of physically contiguous bytes of storage.

SEMAPHORE- a data object which consists of a queue of pointers to waiting processes and a count. Semaphores can be used to synchronize the activities of a set of processes and process to process communication.

SIGNAL- a software interrupt that is global to all partitions.

SOURCE CODE- the original mnemonic or high-level statement versions of a program.

STACK- a data structure which temporarily holds information using a last in first out algorithm.

STACK SEGMENT- specifies the current execution state of the program.

TASK- a routine or machine run. A program may be broken down into several subjobs or tasks to utilize the CPU time more effectively.

VIRTUAL MEMORY- a technique that permits the user to treat secondary (disc) storage as an extension of main memory, thus giving the virtual appearance of a larger memory.



**** ACRONYMNS YOU ALWAYS WANTED TO KNOW ****

AGP- Advanced Graphics Package
ASI- Asynchronous Serial Interface

BCD- Binary Coded Decimal
BPC- Binary Processor Chip

CAM- Content Addressable Memory
CMS-
CP- Current Position (pointer)
CPU- Central Processing Unit
CS- Code Segment
CSMA-
CST- Code Segment Table
CTS- Clear To Send

DAM- Directory Access Method
DB- Data Base (register)
DBM- Data Base Management
DDM- Device Driver Module
DGL- Device-independent Graphics Language
DL- Data Limit (register)
DRT- Device Reference Table
DS- Distributed Systems
DSR- Data Set Ready
DST- Data Segment Table
DTR- Data Terminal Ready

EDS- External Data Segment
EMI- Electro Magnetic Interference

EOT- End Of Transmission

EP- Executive Process

EPP- External Program Pointer

ERS- External Reference Specifications

FA- File Attribute

FIFO- First In First Out

GDS- Global Data Segment

GDU- Graphics Display Units

HI- Human Interface

HIP- Human Interface Process

HPGL- Hewlett-Packard Graphics Language

HPUX- Hewlett-Packard Unix (~~alias Hockey-Pux~~)

ICS- Interrupt Control Stack

IP- Internet Protocol

IPC- Inter Process Communications

IOP- Input Output Processor

IOS- Input Output Switch

IRS- Internal Reference Specifications

ISC- Interface Select Code

ITE- Internal Terminal Emulator

LALIU- Local Area Link Interface Unit

LAN- Local Area Network

LIF- Logical Interchange Format

LIFO- Last In First Out

LVS- Logical View Surface

MCRM- Mapping Content Addressable Memory

MC- Memory Controller

MCB- Memory Controller Bus

MCC- Memory Controller Chip

MPB- Memory Processor Bus

MSI- Mass Storage Is

MSUS- Mass Storage Unit Specifier

MU- Multi-user

NDC- Normalized Device Coordinates

NFT- Network File Transfer

NIH- Not Interface Here

NIU- Network Interface Unit

NVM- Non-Volatile Memory

OS- Operating System

OSI- Open Systems Interconnect

P- Program counter (register)

PA- Primary Address

PB- Program Base (register)

PEDS- Paged External Data Segment

PL- Program Limit (register)

PP- Production Program

PPG- Per Process Global

PSI- Programmable Serial Interface

PTS- Paged Table Segment

Q- stack marker pointer (register)

RAM- Random Access Memory

RFA- Remote File Access

RFI- Radio Frequency Interference
RJE- Remote Job Entry
RP- Run Process
RPM- Remote Process Management
RTC- Real Time Clock
RTS- Request To Send

S- Stack pointer
SA- Secondary Address
SB- Stack Base (register)
SC- Select Code
SCCS- Source Code Control System
SDF- Structured Directory Format
SL- Stack Limit (register)
SOLC- Start Of Line Check (Focus instruction)
SS- Stack Segment
SRM- Shared Resource Manager
STT- Segment Transfer Table
SU- Single-User

TCB- Task Control Block
TCP- Transmission Control Protocol
TSO- Time Sharing Operating system (IBM370)

U-B- Ungermann-Bass
UDU- User Defined Units

VM- Virtual Memory
VT- Virtual Terminal